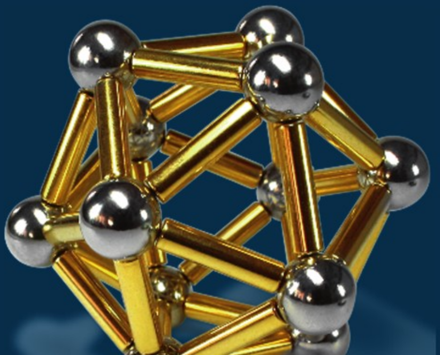
Festschrift

LNCS 7260

Antje Düsterhöft
Meike Klettke
Klaus-Dieter Schewe (Eds.)

Conceptual Modelling and Its Theoretical Foundations

Essays Dedicated to Bernhard Thalheim on the Occasion of His 60th Birthday



Springer

Antje Düsterhöft   Meike Klettke
Klaus-Dieter Schewe (Eds.)

Conceptual Modelling and Its Theoretical Foundations

Essays Dedicated to Bernhard Thalheim
on the Occasion of His 60th Birthday

Springer

Volume Editors

Antje Düsterhöft
University of Applied Sciences Wismar
Philipp Müller Straße, 23952 Wismar, Germany
E-mail: antje.duesterhoeft@hs-wismar.de

Meike Klettke
University of Rostock
Albert-Einstein-Strasse 21, 18051 Rostock, Germany
E-mail: meike@informatik.uni-rostock.de

Klaus-Dieter Schewe
Software Competence Center Hagenberg and
Johannes-Kepler-University Linz
Softwarepark 21, 4232 Hagenberg, Austria
E-mail: kd.schewe@scch.at

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-28278-2 e-ISBN 978-3-642-28279-9
DOI 10.1007/978-3-642-28279-9
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012930755

CR Subject Classification (1998): F.3, D.2, F.4.1, I.2, D.3, H.1

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Over the last decades Bernhard Thalheim has played a major role in the database and conceptual modelling communities. His research interests have spanned a wide spectrum including database dependency theory, object-oriented databases, triggers, abstract state machines, database and information systems design, business processes, and many more.

In March 2012 Bernhard celebrated his 60th birthday. To mark this occasion we invited many of Bernhard's collaborators over the years to contribute to a volume in his honor. This book is the result of these efforts. The collection starts with an academic biography, followed by 20 scientific articles. Many of the contributions also contain a personal tribute based on working experience with Bernhard. The articles cover a broad range of topics, but they still only represent a small fraction of Bernhard's diverse areas of interest.

Each contribution was carefully reviewed by one or two readers. We would like to thank all authors and all reviewers for their contributions and help in producing this book.

We offer this volume to Bernhard in honor of his 60th birthday and in recognition of his many inspiring contributions to the fields of databases, information systems, conceptual modelling and software engineering.

December 2011

Antje Düsterhöft
Meike Klettke
Klaus-Dieter Schewe

Table of Contents

Dedication to a Theory of Modelling

Bernhard Thalheim's Scientific Journey

Antje Düsterhöft[1], Meike Klettke[2], and Klaus-Dieter Schewe[3,4]

[1] University of Applied Sciences Wismar, Germany
antje.duesterhoeft@hs-wismar.de
[2] University of Rostock, Germany
meike@informatik.uni-rostock.de
[3] Software Competence Center Hagenberg, Austria
kd.schewe@scch.at
[4] Johannes-Kepler-University Linz, Austria
kd.schewe@faw.at, kdschewe@acm.org

Bernhard Thalheim's personal website at Christian-Albrechts-University shows a logo that is meant to illustrate the activities of his research group named "Technology of Information Systems". The logo shows a triangle with *Information* in the centre, and *Content*, *Concept* and *Topic* at the three corners. If we were to suggest a logo that illustrates the scientific works and ambitions of Bernhard Thalheim, we would place *Modelling* into the centre of the triangle, and label the corners with *Theory*, *Methods* and *Applications*. So far, Bernhard's scientific life led him from mathematics and pure theoretical computer science over database theory to the design of information systems. What looks on the surface like a movement from one end of computer science to another is de facto the enrichment of a global picture, which ultimately aims at a general theory of modelling. In the sense of the German philosopher Georg Wilhelm Friedrich Hegel Bernhard Thalheim has developed himself as a virtuoso for playing with the dialectic antipodes of theory and practice bringing theoretical results into applications and extracting challenging questions from applications in order to shift scientific knowledge to a higher level as kind of "negated negation". Most recently, he has explicitly declared his ambitions to develop a theory of modelling in an interdisciplinary setting that brings together researchers from mathematics, sciences, economics, engineering and humanities.

Bernhard Thalheim was born on 10th March 1952 in Radebeul in the former German Democratic Republic. With an education that encouraged him to become ambitious he showed his extra-ordinary talents already at school. Consequently, his achievements allowed him to continue studying at a university, though he was first educated as an electrical mechanics. He started to study mathematics at Dresden University of Technology, where he continued as one of the best students in his year, which entitled him to continue with further studies towards a Ph.D. He was selected for research in cybernetics, and received a scholarship to conduct his doctoral studies at Lomonossov University, one of the four elite universities in the former Soviet Union.

So Bernhard faced several real challenges at the same time. He had to learn Russian in very short time, which was probably the subject he most disliked at school. He had to cope with a completely new environment, where knowledge how to circumvent the many little restrictions of the Soviet society was essential, and he had to keep up with high workload, challenging tasks and enormous expectations in one the scientifically most renowned school of the country under the leadership of Andrei Nikolajevitch Kolmogorov. He mastered all these challenges perfectly. He became fluent in Russian, and he adapted easily to life in Moscow. Though contacts of students of different gender was not encouraged by the authorities, he also met his wife Valeria during his study times at Lomonossov. With Valeria he had two sons, Konrad and Klaus, who are his pride and whom he educated sacrificially. For instance, when they were little Bernhard practiced the rule to invent new stories for them on an almost daily basis; later he supported them in their studies and follow-on search for jobs.

Scientifically, he lived up to the expectations of the Kolmogorov school of excellence. For us it seems beyond any imagination to confront a student of the current generation with a task to prepare a seminar talk in a few weeks time with a problem no lesser than to present a novel proof of Turing's incompleteness theorem. At Kolmogorov's school this was normal; for Bernhard it meant very hard work and in particular searching for fragments of such a proof in work of Uspenski.

This challenging – and we should also add very supportive – environment laid the foundations for Bernhard to become the successful researcher we know now. It is still surprising for us to see how much knowledge he has acquired about mathematics and computer science way beyond his own research activities. It came as no surprise that he completed his Ph.D. at Lomonossov University with the top grade for a thesis on completeness and representation conditions for stabile automata. More than that, he established close ties with Russian researchers, which helped him later in his career. In particular, he kept close contact to the statistician Oleg Seleznev, whom he inspired later to apply his knowledge to deep problems in database theory.

Back in east Germany he continued his academic career at Dresden University of Technology as Research Assistant in Computer Science, where he was pushed to slightly change his research orientation. With the immense mathematical knowledge he had gained in his Russian days he turned towards database theory with particular focus on dependency theory. He soon achieved results for several relevant problems. Just to mention two, he discovered an axiomatisation for full join dependencies using a Gentzen style approach, and together with Oleg Seleznev determined bounds for the numbers of shortest keys in relational databases with non-uniform domains. His first book "Dependencies in Relational Databases", published 1991 by Teubner-Verlag, is still a highlight of his works during that time. It is partly a survey on dependency theory and as such a valuable reference for many new research endeavours and partly a research monograph summarising his own achievements in the field.

Bernhard also engaged himself as founder of the "Mathematical Foundations of Databases" (MFDBS) conference, the East-European conference on database theory. The establishment of MFDBS led to several new contacts and long-lasting collaborations, in particular with Janos Demetrovics, the head of SZTAKI, and Gyula Katona, the head of the Alfréd Rényi Institute. It also led to starting research collaboration and friendship with several West-European database researchers, in particular Joachim Biskup and Jan Paredaens.

Naturally, dependency theory in relational databases was also the topic of his habilitation, which he completed 1985 in Dresden. With this he managed to open the door for a continuation as independent researcher, as habilitation was a mandatory prerequisite for becoming professor. In 1989 Bernhard was appointed as professor for theoretical computer science at the University of Rostock, which at that time hosted the second largest department of Computer Science in the German Democratic Republic.

However, before picking up his position in Rostock, Bernhard spent two years as guest professor in Kuwait. Once again this activity required all his adaptation talent to cope with the arabic culture. How do you get a good glass of wine in a country, in which alcohol is completely banned? Bernhard found the solution by turning himself into a respected wine maker, which surprisingly is legal as long as you do not sell the produce. He also turned himself into a perfect bargainer, which is an indispensible gift in societies such as Kuwait. However, he also got experience with negative aspects such as being convicted for causing a car accident because of too cautious driving. Fortunately, he managed to get himself out of punishment.

Scientifically, he started to further widen his scope, first by generalising his results in database theory to new data models, in particular the Entity-Relationship model, then by advancing the field of database design. In Kuwait he developed the database design tool $(DB)^2$, which was based on first versions of his extended Entity-Relationship model. Surprisingly, the tool is still used in several arabic countries and other countries such as Malaysia. His work on database design led to his second book on "Practical Database Design Methodologies", which he wrote together with M. Yaseen and which was published 1989 in Kuwait.

The time in Rostock and also his first years in Cottbus where he moved in 1993 were dominated by his research on the Entity-Relationship model, its theoretical foundations, and its use for the design of information systems. Before Bernhard started his research, the model was smiled at by most database researchers as a rather simple, inexpressive model that has a place at best in sketching database schemata before starting the real design work. Bernhard did not care about these prejudices: the wide use of the model in industrial practice showed that the simplicity of thinking in entities as the key elements about which data should be stored and relationships between them appeals to users who require adequate support from research. So he addressed the weak expressiveness by simply allowing relationships over relationships and structured attributes. Despite other attempts to develop "semantic data models" dedicated to high level database modelling, Bernhard's approach preserved the simplicity of the model, while at the same time giving it a precise mathematical semantics. Furthermore, he shifted dependency theory from the relational model to the extended Entity-Relationship model and extended the theory towards cardinality constraints, path constraints and others.

His third book on "Fundamentals of Entity-Relationship Modeling", however, took a long time to get published. The first version was almost finished in 1992, but featherbrained comments from reviewers, who argued against the need to have higher-order relationships or more than binary relationships, made Bernhard decide to add detailed counter-arguments in the book. So the book was only published in 2000. Bernhard also widened the scope of the book significantly discussing also other developments in the area of database research and showing that his model is ready to take these developments up. In this way he created a true standard reference for the Entity-Relationship model. More recently, he was awarded with the first-ever Peter-Chen-award for his achievements in conceptual modelling and his long-lasting support for the ER conferences.

Another main line of research during his time in Rostock and Cottbus was the advancement of database design. Following the success of his $(DB)^2$ tool he started a much more challenging project, which led to the RADD tool. He integrated natural language processing as a method for developing skeleton schemata first, and he developed a design theory based on design primitives, which were defined by means of graph transformations. Many Ph.D. and Master theses resulted from this research. For the RADD project he could benefit from his Moscow and Kuwait experiences, where the confrontation with different languages such as Russian and Arabic and their pragmatics of usage in everyday life in- and outside his work environment coined his sensibility and affinitity to natural language processing. Consequently, in the sequel he also emerged as a major player and supporter for the NLDB conferences.

However, the years in Rostock and partly also in Cottbus were not all sunny. The German reunion led to a situation, where researchers in the East were all distrusted and even more so when they were successful. Bernhard experienced the incredible injustice that acceptance of a reasonably paid scholarship for his studies in Moscow was suddenly interpreted as proof of exceeded loyalty to the socialist party, whereas academic non-performance was easily excused, if a researcher only pretended to have been oppressed and adopted the new major disposition. Once again excellent researchers were pushed out of their jobs and replaced by lesser colleagues from the West.

Bernhard Thalheim survived these humiliations, but they left an invisible mark on him, as the former airiness with which he initiated scientific collaborations was gone. Nonetheless, he was one of the first East-German researchers who established collaboration ties with colleagues in the West. Bernhard's conference MFDBS was merged with the International Conference on Database Theory (ICDT), he joined for a couple of years the executive committee of the GI standing working group on foundations of information systems, and he became the one to be contacted first, when East-West collaboration was sought.

On the scientific side, Bernhard had to face another challenge, which he perceived as deterioration of computer science culture. His fundamental results on database theory, modelling and design were not picked up by the majority as expected. Not only few researchers claiming to work in the area of conceptual modelling had adopted the luxury attitude to ignore his results and to still hold up claims that he had proven to be wrong. One example was the use of triggers for consistency enforcement. He suffered being surrounded by many researchers who declared themselves "practical", but had never seen a real application. Instead, "practicality" merely meant the absence of theory, which Bernhard as a learnt theoretician could not accept. Still Bernhard kept his sense of humour, and he turned his experiences into satirical lectures and essays, which are a pleasure to read.

Bernhard himself had left since long the realm of pure theoretical computer science. He had not only established himself as an expert in database and information systems design with deep roots in theory, he had initiated many collaborations with industry including large companies such as Volkswagen, SAP, Vattenfall and others. So he knew what he was talking about when he explained what according to him would be the real research challenges, and in addition, he kept the ties with theory. His well-frequented practitioner forum in Cottbus was a proof of acceptance of his expertise by industrial users.

Around 1997 the world-wide web began to influence Bernhard's work, and he started another line of research, which soon turned out to be just another extension of the work he had already done before. He addressed web information systems, which were originally inspired by the desire to set up a city information system for inhabitants and tourists. From the very beginning usability for the end-user was a key issue: the system should be usuable without training by everyone including elderly citizens who would not like to adopt new fancy technology, and the system should not only run on the web, but also exploit videotext on normal television. Over the years the research has led to a substantial number of scientific results on the design of web information systems, though the book publication summarising all the achievements is still awaiting completion and publication.

There are many who attempted to get a hold on the design of web information systems and to bring order into this area that was easily occupied with many adhoc solutions without deeper scientific grounding. What made Bernhard's ideas special is that he emphasised the needs of the users as starting point for design, while the main trend was to focus on content and presentation. Based on his knowledge he had obtained from his classical school education, he drew the analogy to story telling and consequently to classical drama and the successful methods in movie production. The idea of "storyboarding" was born and later formalised. What made Bernhard's approach unique is that he created it in close connection with more than thirty large application projects comprising other city information sites, e-government and e-learning. More recently he took the storyboarding approach further addressing pragmatics in web information systems, thus investigating systematically what a particular design would mean to users.

We have always experienced Bernhard Thalheim as a visionary person. He has the ability to outline a big picture, where others would only deal with small parts, and he suffered from the simple-mindedness around him. His big pictures were understood by other collaborators like Egon Börger, who like Bernhard understood that the big and unconventional ideas are the main drivers of progress and not the niggling on the detail. Over many years of collaboration with Bernhard we had often corrected little mistakes in his formalisations of detail, but nonetheless his great ideas and his enthusiasm have always been very inspiring. He has also been honoured for this by his appointment as honourary Kolmogorov professor at Moscow State University.

Bernhard's scientific works can almost all be contributed to the area of conceptual modelling. Like no other he created a deep understanding of the data involved, the functionality, the context, and the user needs. His work has proven to be useful for applications, and it is grounded in theoretical foundations. Recently, Bernhard has revealed his plans for the next decade, which are to approach a general theory of modelling that draws not only on information systems, but also borrows knowledge about modelling from various other disciplines. We are confident that he will bring also this endeavour to success, and we wish him the best for it including the necessary scientific and financial support. We are looking forward to the time, where his dedication to developing such a theory of modelling will produce the fruits and show their practical usefulness.

What about Constraints in RDF?

Jan Paredaens

University of Antwerp, Belgium
jan.paredaens@ua.ac.be

1 Dear Bernhard

Dresden, January 19-23, 1789, sorry 1987. The first MFDBS, Mathematical Fundamentals of Database Systems. There we met for the first time. To be still more precise, I arrived with my car on Sunday January 18, around 5 pm. It could have been 5.30 pm. This is the exact point in time we met. Bernhard, it is exactly 25 years ago. Another reason to celebrate! That meeting was the first one in a long list that is surely not complete: Dresden, Berlin, Antwerp, New Jersey, Antwerp, Rostock, Eindhoven, Visegrad, Antwerp, Sofia, ... Kuwait. Indeed I visited you in Kuwait the week of November 9, 1989. I still remember that on that evening I saw, with glass of wine that you produced yourself in your apartment, on your TV set how the Berlin Wall felt down. An historical moment!!!

In preparing this contribution, I realized that you have more than 200 papers in DBLP. This is fantastic! But not only the quantity is important, also the quality and the number of different subjects and areas: dependencies, constraints, normal forms, null values, data models, database design, object oriented databases, conceptual models, semantics of databases, data warehouses, OLAP, database schemes, Web information systems, Web based learning, component-driven engineering of database applications, intention-driven screenography (what the hell it may be), even a fixed-point query language for XML and this list is again far from complete.

Also the number of coauthors, I am proud to be one of them! Bernhard we have 9 papers together! This makes me think of our Erdos number. What is your Erdos number? Mine is 5 to the best of my knowledge. But if yours is smaller than 4, it makes mine better!

Bernhard, you wrote a lot of papers about dependencies and constraints. You are surely the main specialist in the world who knows everything about constraints and dependencies in the relational database model. Even your first paper [11] in 1984 was on constraints. As you know, a constraint is a property of the scheme of a database that has to be satisfied by every instance of the database.

Dependencies and constraints are very important in the relational database world mainly because

- they are the formal foundation for some very important notions, like keys and referential integrity;
- they play a crucial role in the design of a database and the decomposition of relations;

- they can help us to verify whether the content of the database contains correct information; more specifically we can use them to detect some errors in the tables;
- they are just interesting properties that help the user to understand the meaning of the information in the database.

As far as I know or I can find in the literature, Bernhard, I do not think that you have studied RDF (Resource Description Framework) or properties of RDF. Unlike standard autonomous relational databases, data in semantic web repositories is meant to be linked to other data sources available on the web by means of universal identifiers (usually in the form of Uniform Resource Identifiers or URIs) and stored in schema-less repositories. Towards the instantiation and support of this vision, the World Wide Web Consortium (W3C) has proposed the recommendation of the RDF data model and the SPARQL query language [10].

It is not at all easy to understand this complete data model. The current status of RDF can be found on [7]. But this link contains hundreds of pages defining the concepts, the abstract syntax, the RDF semantics, a primer, the vocabulary, the RDF schema, RDF/XML Syntax specification and other material. Bref, too long and too complex to study from a more fundamental and theoretical point of view. But, isn't this also the case for the relational model? Indeed, the relational model can be defined in a few pages, while the documentation of the Oracle Database 11g Release 2, also contains hundreds of pages.

So we first define in a few lines the RDF-model. Then we ask ourselves what a constraint is in the context of RDF, and why constraints are important. It is not because "they play a crucial role in the design of RDF and the decomposition of RDF". But the three other reasons of the above list still hold:

- they are the formal foundation for some very important notions, like RDFS;
- they can help us to verify whether the content of an RDF-document or RDF-resource contains correct information; more specifically we can use them to detect some errors;
- they are just interesting properties that help the user to understand the meaning of the information in an RDF-resource.

We have been investigating in ADReM, my research group in the University of Antwerp, dependencies and constraints in RDF, during recent months [1]. A number of colleagues, also German colleagues, have studied constraints in RDF [2,3,4,5,1]. I hope to give you an overview of some interesting constraints in RDF, by defining them formally, giving examples and a motivation for their use. Clearly, this is not a complete overview, nor do I give here more results and/or theorems, algorithms and techniques, mainly because they are still under investigation. My goal is that the reader in general, but you Bernhard in particular, will understand the importance of constraints in RDF and you and others will contribute in understanding their properties, their role and techniques.

Let's start with the interesting part!

2 The RDF-Model

The semantic web idea envisions the web as a large repository of semantic information that can be accessed and queried by both, machines and human beings. The RDF data model is similar to classic conceptual modeling approaches such as Entity-Relationship or Class diagrams, as it is based upon the idea of making statements about resources (in particular Web resources) in the form of subject-property-object expressions [8]. These expressions are known as triples in RDF terminology. The subject denotes the resource, and the property denotes traits or aspects of the resource and expresses a relationship between the subject and the object. RDF is an abstract model with several serialization formats (i.e., file formats), and so the particular way in which a resource or triple is encoded varies from format to format.

The W3C published a specification of RDF's data model and XML syntax as a Recommendation in 1999. Work then began on a new version that was published as a set of related specifications in 2004 [8,9].

RDF-resources (documents or databases) arrange data in simple triples of *subject*, *property* and *object*. Triples represent assertions about certain predefined domains consisting of *resources*. Resources are identified by URIs. The semantics of (s, p, o) is that "the subject s has property p with value o". Note that a subject can have a same property with many values, which is represented by many triples. RDF-resources are often conceptualized as defining a directed labeled graph, where subjects and objects are nodes and the property represents a directed edge between the subject and the object. From a logical stand, the W3C defines RDF as "...[an] assertional logic, in which each triple expresses a simple proposition. This imposes a fairly strict monotonic discipline on the language, so that it cannot express closed-world assumptions, local default preferences, and several other commonly used non-monotonic constructs". While RDF-resources consist of simple triples, they cannot be regarded as merely a ternary relation of the relational model. Indeed, in RDF there is only one set of triples; RDF triples are heterogeneous entities and as such semantically less structured than in the relational model. Observe that a property in a triple can be the subject or the object in another (or the same) triple. As a consequence the languages for RDF are more pattern-based than in the relational counterpart.

Throughout this section and the rest of the paper we make references to Figure 1, which depicts an RDF-resource that stores information about publications, authors and characters of books.

Subject and object are nodes of the graph, while edges represent properties. For instance, triple $(\mathbf{Literaria}, \mathbf{sp}, \mathbf{Magazine})$ states that subject **Literaria** is a subproperty (**sp**) of the object **Magazine**. Likewise, triple $(\mathbf{Paper}, \mathbf{sp}, \mathbf{Journal})$ asserts that subject **Paper** is a subproperty of the object **Journal**. Observe that, unlike the case of the relation model, a resource can be both a subject and a property. Indeed, in Figure 1 it is the case that **Don_Quixote** is the property in triples $(\mathbf{Sancho}, \mathbf{Don\_Quixote}, \mathbf{Sanchica})$ and $(\mathbf{Sancho}, \mathbf{Don\_Quixote}, \mathbf{Rocinante})$, and the subject in $(\mathbf{Don\_Quixote}, \mathbf{sp}, \mathbf{Book})$. Similarly, **Literaria** is the property in triple $(\mathbf{Metaphysics}, \mathbf{Literaria}, \mathbf{Ethics})$, the object in triple

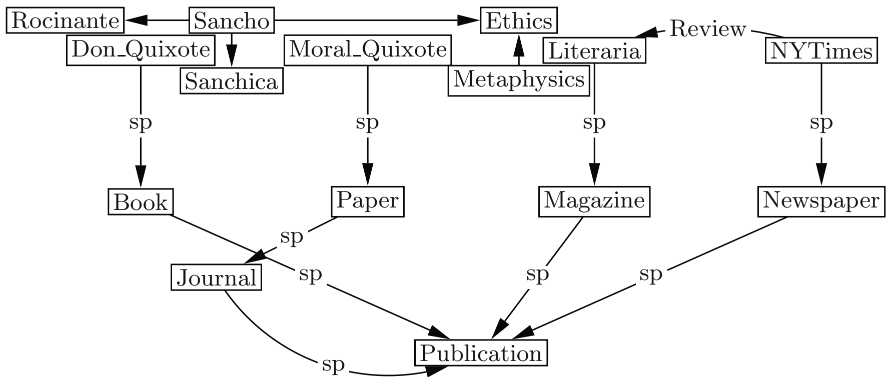


Fig. 1. RDF-Graph representing information about different types of publications

(**NYTimes**, **Review**, **Literaria**) and the subject in (**Literaria**, **sp**, **Magazine**). This is one of the distinctive features of the RDF data model which makes the definition of constraints into RDF not just a trivial migration of constraints from the relational domain.

Hence an RDF-resource is just a finite set of triples (subject,property,object). It is called in the literature *an RDF-graph*, although it is not formally a graph, since we can have edges between edges, or edges between a node and another edge, which is obviously impossible in a classical graph.

In what follows, we use the following vocabulary:

- $\mathcal{U}$, is an infinite set of URI's;
- $\mathcal{L}$, is an infinite set of literals;
- V, an infinite set of variables denoted by prefixing them by \$.

V, $\mathcal{U}$ and $\mathcal{L}$ are pairwise disjoint.

Definition 1 (RDF-graph). *An* RDF-graph $\mathcal{G}$ *is a finite set of triples* (s, p, o), *subject, property, object,* $s, o \in \mathcal{U} \cup \mathcal{L}$, $p \in \mathcal{U}$[1].

Figure 2 contains 14 triples.

We do not consider here *blank nodes.* This is not because they are not important! Indeed, they are one of the main concepts in the RDF-model but we do not consider them here because their presence complicates the definitions of constraints on the one hand and, to be honest Bernhard, they are still under investigation, the moment I write this sentence.

3 Equality Generating and Triple Generating Constraints

Here there is really nothing new! We know what equality generating and tuple generating constraints are in the relational model. Since an RDF-graph is

[1] We denote $\mathcal{U}_{\mathcal{G}}$ (resp. $\mathcal{L}_{\mathcal{G}}$) for the set of elements of $\mathcal{U}$ (resp. $\mathcal{L}$) that occur in $\mathcal{G}$.

formally nothing else than an untyped ternary table we obtain the analogous constraints for RDF-graphs. To be complete, and to define the notation we give here their formal definitions.

A *term* is an element of $V \cup \mathcal{U} \cup \mathcal{L}$.

In the following definition S is a finite set of terms. It will play the role of pattern for the parts of the RDF-graph we are interested in.

Definition 2 (Embedding of a set of triples of terms in an RDF-graph $\mathcal{G}$). *An embedding of a set S of triples of terms in an RDF-graph $\mathcal{G}$ is a total function $e : V_S \cup \mathcal{U}_S \cup \mathcal{L}_S \rightarrow \mathcal{U}_\mathcal{G} \cup \mathcal{L}_\mathcal{G}$, such that*

- *$e(u) = u$, for each $u \in \mathcal{U}_S$;*
- *$e(l) = l$, for each $l \in \mathcal{L}_S$;*
- *If $(t_1, t_2, t_3) \in S$ then $(e(t_1), e(t_2), e(t_3)) \in \mathcal{G}$;*

Definition 3 (EGC). *An equality generating constraint is a pair (S, E)*[2]*, where*

- *S is a finite set of triples of terms;*
- *E is a finite set of equalities, each of the form $(t_1 = t_2)$, with $t_1, t_2 \in V_S \cup \mathcal{U} \cup \mathcal{L}$.*

Definition 4 (EGC satisfaction). *An RDF-graph $\mathcal{G}$ satisfies the EGC (S, E) iff for every embedding e of S in $\mathcal{G}$ and every $(t_1 = t_2) \in E$ holds that $e(t_1) = e(t_2)$.*

Example 1. Figure 1. satisfies:

- $\mathcal{C}_1 = (\{(\mathbf{Sancho}, \mathbf{Don\_Quixote}, \$y), (\mathbf{Don\_Quixote}, \mathbf{sp}, \$t)\}, \{(\$t = \mathbf{Book})\})$
- $\mathcal{C}_2 = (\{(\mathbf{Book}, \mathbf{sp}, \$y), (\mathbf{Journal}, \mathbf{sp}, \$z)\}, \{(\$y = \$z)\})$

Figure 1. does not satisfy:

- $\mathcal{C}_3 = (\{(\$x, \mathbf{sp}, \$z), (\$z, \mathbf{sp}, \$u)\}, \{(\$u = \mathbf{Publication})\})$
- $\mathcal{C}_4 = (\{(\$x, \mathbf{Don\_Quixote}, \$y)\}, \{(\$y = \mathbf{Sanchica})\})$
- $\mathcal{C}_5 = (\{(\$x, \mathbf{sp}, \$z), (\$v, \$u, \$x)\}, \{(\$u = \mathbf{sp})\})$

In [1] we define a set of deductive axioms for deriving EGCs.
Let $\mathcal{EC}$ be an arbitrary set of EGCs. We define in a syntactical way, by Rules 0-8 when $\mathcal{EC} \vdash (S, E)$, and we prove that (S, E) is a logical consequence of $\mathcal{EC}$ iff $\mathcal{EC} \vdash (S, E)$.

The rules that define $\vdash$ are :

Rule 0 : $\mathcal{EC} \vdash (S, E)$, for every $(S, E) \in \mathcal{EC}$;
Rule 1 : $\mathcal{EC} \vdash (S, \{(t = t)\})$, for every finite set S of triples of terms and $t \in V_S \cup \mathcal{U}_S \cup \mathcal{L}_S$;

[2] We denote $\mathcal{U}_S$ (resp. $\mathcal{L}_S$, V_S) for the set of elements of $\mathcal{U}$ (resp. $\mathcal{L}$, V) that occur in S.

Rule 2 : $\mathcal{EC} \vdash (S, \{(t_1 = t_2)\})$ implies $\mathcal{EC} \vdash (S, \{(t_2 = t_1)\})$;
Rule 3 : $\mathcal{EC} \vdash (S, \{(t_1 = t_2), (t_2 = t_3)\}$ implies $\mathcal{EC} \vdash (S, \{(t_1 = t_3)\}$;
Rule 4 : $\mathcal{EC} \vdash (S, E)$ and $E_1 \subseteq E$ implies $\mathcal{EC} \vdash (S, E_1)$;
Rule 5 : $\mathcal{EC} \vdash (S, E_1)$ and $\mathcal{EC} \vdash (S, E_2)$ implies $\mathcal{EC} \vdash (S, E_1 \cup E_2)$;
Rule 6 : $\mathcal{EC} \vdash (S, E)$ and h is a homomorphism from S in S_1 implies $\mathcal{EC} \vdash (S_1, h(E))$;
Rule 7 : $\mathcal{EC} \vdash (S, \{(t = t')\})$ and $\mathcal{EC} \vdash (\phi_{t \hookleftarrow t'}(S), E)$ implies $\mathcal{EC} \vdash (S, E)$;
Rule 8 : $\mathcal{EC} \vdash (S, \{(a = b)\})$ for $a, b \in \mathcal{U}_S \cup \mathcal{L}_S$ and $a \neq b$ implies $\mathcal{EC} \vdash (S, E)$ for all possible E.

Theorem 1. *[1] The Rules 0-8 are sound, independent and complete.*

Definition 5 (TGC). *A triple generating constraint is a pair (S, S'), where both S and S' are finite sets of triples of terms.*

Definition 6 (TGC satisfaction). *An RDF-graph $\mathcal{G}$ satisfies the TGC (S, S') iff for every embedding e of S in $\mathcal{G}$ there exists an embedding e' of $(S \cup S')$ in $\mathcal{G}$ such that $(e(t_1), e(t_2), e(t_3)) = (e'(t_1), e'(t_2), e'(t_3))$ for every $(t_1, t_2, t_3) \in S$.*

Example 2. Figure 1. satisfies:

- $\mathcal{C}_6 = (\{(\mathbf{Paper}, \mathbf{sp}, \mathbf{Journal})\}, \{(\mathbf{NYTimes}, \mathbf{sp}, \mathbf{Newspaper})\})$
- $\mathcal{C}_7 = (\{(\mathbf{Moral\_Quixote}, \mathbf{sp}, \$y), (\$y, \mathbf{sp}, \mathbf{Journal})\}, \{(\mathbf{Journal}, \mathbf{sp}, \mathbf{Publication})\})$
- $\mathcal{C}_8 = (\{(\mathbf{Don\_Quixote}, \mathbf{sp}, \mathbf{Book})\}, \{(\mathbf{NYTimes}, \mathbf{sp}, \$y), (\$y, \mathbf{sp}, \mathbf{Publication})\})$
- $\mathcal{C}_9 = (\{(\mathbf{Paper}, \$x, \mathbf{Publication})\}, \{(\$y, \mathbf{sp}, \mathbf{Ethics})\})$

Figure 1. does not satisfy:

- $\mathcal{C}_{10} = (\{(\mathbf{Paper}, \mathbf{sp}, \mathbf{Journal})\}, \{(\mathbf{NYTimes}, \mathbf{sp}, \mathbf{Publication})\})$.

4 RDFS

An RDF Schema (abbreviated as RDFS) is an extensible knowledge representation language, providing basic elements for the description of ontologies, called RDF vocabularies, intended to structure RDF-resources. The first version was published by W3C in April 1998, and the final W3C recommendation was released in February 2004 [6].

Basically RDFS defines a number of special RDF-properties, that are frequently used in RDF-resources. In this section we will mention some of these RDF-properties and formally define their semantics and inference rules by triple generating constraints. This illustrates the use of the triple generating constraints introduced above.

The RDF-properties of RDFS that we will consider are *subproperty*, *subclass*, *typing*, *domain* and *range*. Consider the RDF-subjects or RDF-objects (that by the way can also be RDF-properties) S_1, S_2, S_3, the RDF-properties P_1, P_2, P_3, the classes of objects or subjects (or types) C_1, C_2 and C_3.

Subproperty

The RDFS subproperty is denoted by **sp**. $(P_1, \mathbf{sp}, P_2)$ indicates that P_1 is a subproperty of P_2. The semantics of **sp** says that

- if P_1 is a subproperty of P_2 and P_2 is a subproperty of P_3 then P_1 is a subproperty of P_3; this is expressed by the triple generating constraint:

$$(\{(\$p_1, \mathbf{sp}, \$p_2), (\$p_2, \mathbf{sp}, \$p_3)\}, \{(\$p_1, \mathbf{sp}, \$p_3)\})$$

- if S_1 has as property P_1 with value S_2 and P_1 is a subproperty of P_2 then S_1 has also as property P_2 with value S_2, formally:

$$(\{(\$s_1, \$p_1, \$s_2), (\$p_1, \mathbf{sp}, \$p_2)\}, \{(\$s_1, \$p_2, \$s_2)\})$$

- every property between a subject and an object is a subproperty of itself, formally:

$$(\{(\$s_1, \$p, \$s_2)\}, \{(\$p, \mathbf{sp}, \$p)\})$$

- if P_1 is a subproperty of P_2 then P_1 and P_2 are each a subproperty of itself, formally:

$$(\{(\$p_1, \mathbf{sp}, \$p_2)\}, \{(\$p_1, \mathbf{sp}, \$p_1), (\$p_2, \mathbf{sp}, \$p_2)\})$$

Subclass

The RDFS subclass is denoted by **sc**. $(C_1, \mathbf{sc}, C_2)$ indicates that C_1 is a subclass of C_2. The semantics of **sc** says that

- if C_1 is a subclass of C_2 and C_2 is a subclass of C_3 then C_1 is a subclass of C_3, formally:

$$(\{(\$c_1, \mathbf{sc}, \$c_2), (\$c_2, \mathbf{sc}, \$c_3)\}, \{(\$c_1, \mathbf{sc}, \$c_3)\})$$

- if C_1 is a subclass of C_2 then C_1 and C_2 are each a subclass of itself, formally:

$$(\{(\$c_1, \mathbf{sc}, \$c_2)\}, \{(\$c_1, \mathbf{sc}, \$c_1), (\$c_2, \mathbf{sc}, \$c_2)\})$$

Typing, Domain, Range

The RDFS typing is denoted by **type**, the domain of a type by **dom**, its range by **range**. $(S_1, \mathbf{type}, C_1)$ indicates that C_1 is a type of S_1. $(P_1, \mathbf{dom}, C_1)$ indicates that C_1 is the domain of P_1. $(P_1, \mathbf{range}, C_1)$ indicates that C_1 is the range of P_1. Their semantics says that

- if C_1 is a subclass of C_2 and C_1 is a type of S_1 than C_2 is also a type of S_1, formally:

$$(\{(\$c_1, \mathbf{sc}, \$c_2), (\$s_1, \mathbf{type}, \$c_1)\}, \{(\$s_1, \mathbf{type}, \$c_2)\})$$

- if C_1 is the domain P_1, and S_1 has property P_1, then C_1 is a type of S_1, formally:

$$(\{(\$p_1, \mathbf{dom}, \$c_1), (\$s_1, \$p_1, \$s_2)\}, \{(\$s_1, \mathbf{type}, \$c_1)\})$$

- if C_1 is the range of the P_1, and S_2 is a value of the property P_1, then C_1 is a type of S_1, formally:

$$(\{(\$p_1, \mathbf{range}, \$c_1), (\$s_1, \$p_1, \$s_2)\}, \{(\$s_2, \mathbf{type}, \$c_1)\})$$

- if C_1 is the domain of P_1, P_2 is a subproperty of P_1 and S_1 has property P_2 with value S_2 then C_1 is a type of S_1

$$(\{\$p_1, \mathbf{dom}, \$c_1), (\$p_2, \mathbf{sp}, \$p_1), (\$s_1, \$p_2, \$s_2)\}, \{(\$s_1, \mathbf{type}, \$c_1)\})$$

- if C_1 is the range of P_1, P_2 is a subproperty of P_1 and S_1 has property P_2 with value S_2 then C_1 is a type of S_2

$$(\{\$p_1, \mathbf{range}, \$c_1), (\$p_2, \mathbf{sp}, \$p_1), (\$s_1, \$p_2, \$s_2)\}, \{(\$s_2, \mathbf{type}, \$c_1)\})$$

- **sp**, **sc**, **dom**, **range**, **type** are each a subproperty of itself, formally:

$$(\emptyset, \{\mathbf{v}, \mathbf{sp}, \mathbf{v})\}), \mathbf{v} \in \{\mathbf{sp}, \mathbf{sc}, \mathbf{dom}, \mathbf{range}, \mathbf{type}\}$$

- every property that has a domain or a range is a subproperty of itself, formally:

$$(\{(\$p, \mathbf{v}, \$y)\}, \{(\$p, \mathbf{sp}, \$p)\}), \mathbf{v} \in \{\mathbf{dom}, \mathbf{range}\}$$

- every class that is the domain, the range or a type is a subclass of itself, formally:

$$(\{(\$x, \mathbf{v}, \$c)\}, \{(\$c, \mathbf{sc}, \$c)\}), \mathbf{v} \in \{\mathbf{dom}, \mathbf{range}, \mathbf{type}\}$$

5 Functional Constraints

The functional constraints are similar to the functional dependencies. However there are some major differences. They indicates that if some subjects, properties or objects are equal in an RDF-graph then some other subjects, properties or objects have to be equal.

Definition 7 (FC). *A functional constraint is a pair* $(S, L \to R)$, *where*

- S *is a finite set of triples of terms;*
- $L, R \subseteq V_S$.

Definition 8 (FC satisfaction). *An RDF-graph* $\mathcal{G}$ *satisfies the FC* $(S, L \to R)$ *iff for every two embeddings of* S *in* $\mathcal{G}$ *that coincide*[3] *on the variables of* L, *they also coincide on the variables of* R.

Example 3. Figure 1. satisfies:

- $\mathcal{C}_{11} = (\{(\$x, \mathbf{sp}, \$y)\}, \{\$x\} \to \{\$y\})$
- $\mathcal{C}_{12} = (\{(\$x, \mathbf{sp}, \$y), (\$y, \mathbf{sp}, \$z)\}, \{\$x, \$z\} \to \{\$y\})$
- $\mathcal{C}_{13} = (\{(\mathbf{Sancho}, \$x, \$y), (\$x, \mathbf{sp}, \$z)\}, \emptyset \to \{\$z\})$

[3] Two embeddings e and e' coincide on a variable $\$v$ iff $e(\$v) = e'(\$v)$.

Figure 1. does not satisfy:

- $\mathcal{C}_{14} = (\{(\$x, \$y, \$z), (\$y, \mathbf{sp}, \$y_1), (\$y_1, \mathbf{sp}, \$y_2)\}, \emptyset \to \{\$y_2\})$
- $\mathcal{C}_{15} = (\{(\$x, \mathbf{sp}, \$y)\}, \{\$y\} \to \{\$x\})$
- $\mathcal{C}_{16} = (\{(\$x, \mathbf{Don\_Quixote}, \$y)\}, \{\$x\} \to \{\$y\})$
- $\mathcal{C}_{17} = (\{(\$x, \$y, \$z)\}, \{\$x\} \to \{\$y\})$
- $\mathcal{C}_{18} = (\{(\$x, \$y, \$z)\}, \{\$y\} \to \{\$x\})$
- $\mathcal{C}_{19} = (\{(\$x, \$y, \$z), (\$y, \mathbf{sp}, \$y_1)\}, \{\$x\} \to \{\$y, \$y_1\})$

Note that every functional constraint is equivalent with an equality generating constraint, but not vice-versa. This is illustrated by the functional constraint $\mathcal{C}_{20} = (\{(a, \$x, \$y)\}, \{\$x\} \to \{\$y\})$ which is equivalent with the equality generating constraint $\mathcal{C}_{21} = (\{(a, \$x_1, \$y_1), (a, \$x_1, \$y_2)\}, \{(\$y_1 = \$y_2)\})$, but on the other hand the equality constraint $\mathcal{C}_{22} = (\{(\$x, \$y, \$z)\}, \{(\$x = \$y)\}$ is equivalent with no finite set of functional constraints, since the graph $\{(a, b, c)\}$ does not satisfy $\mathcal{C}_{22}$ but it satisfies all FCs.

6 Forbidding Constraints

While equality generating and functional constraint require properties for all possible embeddings, forbidding constraints forbid some embeddings.

Definition 9 (FBC). *A forbidding constraint has the form (S), where S is a finite set of triples of terms.*

Definition 10 (FBC satisfaction). *An RDF-graph $\mathcal{G}$ satisfies the FBC (S) iff there is no embedding of S in $\mathcal{G}$.*

Example 4. Figure 1. satisfies:

- $\mathcal{C}_{23} = (\{(\$x, \mathbf{sp}, \$x)\})$

Figure 1. does not satisfy:

- $\mathcal{C}_{24} = (\{(\$x, \mathbf{sp}, \$y)\})$

Forbidding constraints are rather independent from equality generating constraints. Indeed the forbidding constraint $\{(a, \$x, \$y)\}$ nor its negation is equivalent with a finite set of equality generating constraints. Furthermore the equality generating constraint $\mathcal{C}_{25} = (\{(\$x, \$y, \$z)\}, \{(\$x = \$y), (\$x = \$z), (\$x = a)\}$ is only satisfied by the graph $\{(a, a, a)\}$. So $\mathcal{C}_{25}$ nor its negation is equivalent to a finite set of forbidding constraints.

Forbidding constraints are also rather independent from functional constraints. Indeed, some forbidding constraints nor their negation are equivalent with a finite set of functional constraints, otherwise they would be equivalent with a finite set of equality generating constraints. Furthermore the functional constraint $\mathcal{C}_{26} = (\{(\$x, \$y, \$z)\}, \emptyset \to \{\$x, \$y, \$z\})$ is only satisfied by the singleton graphs. So $\mathcal{C}_{26}$ nor its negation is equivalent to a finite set of forbidding.

In many RDF-resources "the number of levels of properties" is limited by a constant. For instance consider **person**s that have **account**s on **bank**s, each **bank** has an **address** and a **list of employees**, that can be found on separate **URL**s, the **address** and the **list of employees** are subproperties of **info**. An example of such an RDF-resource is given in Figure 2. The level of properties is

$$(\mathbf{per_1}, \mathbf{ban_1}, \mathbf{acc_1}), (\mathbf{per_1}, \mathbf{ban_2}, \mathbf{acc_2}), (\mathbf{ban_1}, \mathbf{add}, \mathbf{t_1}), (\mathbf{ban_1}, \mathbf{loe}, \mathbf{URL_1}),$$

$$(\mathbf{ban_2}, \mathbf{loe}, \mathbf{URL_2}), (\mathbf{add}, \mathbf{sp}, \mathbf{info}), (\mathbf{loe}, \mathbf{sp}, \mathbf{info})$$

Fig. 2. Example for levels of properties

here 3 since $\mathbf{ban_1}$ is a property, it has a property **add** and **add** has a property **sp**. More formally, the level of properties of an RDF-graph $\mathcal{G}$ is the greatest number n such that $\exists \mathbf{a_0}, \ldots, \mathbf{a_n}, \mathbf{b_1}, \ldots, \mathbf{b_n}$ with

$$\{(\mathbf{b_0}, \mathbf{a_1}, \mathbf{b_1})\} \cup \{(\mathbf{a_{i-1}}, \mathbf{a_i}, \mathbf{b_i}) \mid 1 < i \leq n\} \subseteq \mathcal{G}$$

Let us express now that the level of properties of an RDF-graph may not exceed 2, this is expressed by the following forbidding constraint:

$$\{(\$b_0, \$a_1, \$b_1), (\$a_1, \$a_2, \$b_2), (\$a_2, \$a_3, \$b_3)\}$$

7 Enumerable Constraints

Let us call the constraints that we defined above *finite constraints* since all contain one or two **fixed finite** sets of triples of terms. But suppose we want to specify a cycle or a non-regular pattern. To express this we can use so-called *e-sets*, that are sets with one numerical parameter.

Every instantiation of an e-set $S(n)$, for a particular value of $n \geq 0$, is a finite set. Let us give some examples of indexed sets.

Example 5. Indexed Sets

- $A(n) = \{2, 4, \ldots, 2 * n\}$; $A(5) = \{2, 4, 6, 8, 10\}$ and $A(0) = \emptyset$ are two instanstations of A;
- $B(n) = \{\mathbf{a}, \mathbf{b}, \$x_0, \ldots, \$x_n\}$;
- $C(n) = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$;
- $C(0) = C(24) = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$;
- $D(n) = \emptyset$;
- $E(n) = \{(\$x_0, \$p_0, \$x_1), \ldots, (\$x_{2.n-1}, \$p_{2.n-1}, \$x_{2.n})\}$, which specifies a list of even length of objects and subjects;
- $F(n) = \{(\$x_n, \$p_n, \$x_0), (\$x_0, \$p_0, \$x_1), \ldots, (\$x_{n-1}, \$p_{n-1}, \$x_n)\}$, which specifies a cycle of objects and subjects;

- $G(n) = \{(\mathbf{a}, \$p_0, \$x_1), (\$x_1, \$p_1, \$x_2), \ldots, (\$x_{n-1}, \$p_{n-1}, \mathbf{b})\}$, which specifies a sequence of objects and subjects that starts with **a** and ends with **b**;
- $H(n) = \{(\$p_0 = \mathbf{a}, \ldots, \$p_n = \mathbf{a}\}$;

If in a finite constraint, S (S',E,L,R) is replaced by an indexed-set, we call it an *enumerable constraint*. An RDF-graph $\mathcal{G}$ satisfies an enumerable constraint $(S(n), E)$ (resp. $(S(n), S'(n))$, $(S(n), L(n) \to R(n))$, $(S(n)))$ if for every non-negative value of v $\mathcal{G}$ satisfies all the finite constraints $(S(v), E(v))$ (resp. $(S(v), S'(v))$, $(S(v), L(v) \to R(v))$, $(S(v)))$.

A few examples can clarify this notion.

Example 6. Enumerable Constraints

- $\mathcal{C}_{27} = (F(n))$, which is a forbidding constraint that forbids a cycle of objects and subjects;
- $\mathcal{C}_{28} = (F(n), H(n))$, which is an equality generating constraint that verifies whether all the properties in a cycle are equal to **a**;
- $\mathcal{C}_{29} = (G(n), \emptyset \to \{\$x_1\})$, which is a functional constraint that verifies whether the second objects of two sequences, that start with **a**, end with **b** and have equal length, are equal;
- $\mathcal{C}_{30} = (\{(\$p_{n+1}, \$p_0, \$x_0), (\$p_0, \$p_1, \$x_1), \ldots, (\$p_n, \$p_{n+1}, \$x_{n+1})\})$ is an interesting forbidding constraint that, I guess, all RDF-graphs in concrete applications satisfy. It forbids cycles in the levels of property.

8 Conclusion

We introduced in this paper some interesting definitions of classes of constraints for RDF-resources. There are surely more. There are a lot of challenges: Which are the interesting constraints for RDF-resources, what can we say about their deduction mechanisms, what is the complexity to verify them, what is their relationship with SPARQL, what is the semantics of blank nodes in all this, can we deduce some interesting practical and theoretical results from them, etc.

It is our hope, dear Bernhard and other colleagues, that this contribution opens a new world of research, that we will understand better the above constraints and their properties, that we can involve other typical RDF features as blank nodes and RDFS, that we will discover other new interesting RDF-constraints whose counterpart is not discussed in the relational model, because of not interesting in that context and that Bernhard and I, very old but still clever, will lean in our rocking chair, reading new and fascinating results on RDF-constraints.

Bernhard, so long but not farewell!

Acknowledgment. Alvaro Cortes, a postdoc at the University of Antwerp, also contributed to this project.

References

1. Akhtar, W., Cortés-Calabuig, Á., Paredaens, J.: Constraints in RDF. In: Schewe, K.-D., Thalheim, B. (eds.) SDKB 2010. LNCS, vol. 6834, pp. 23–39. Springer, Heidelberg (2011)
2. Arenas, M., Gutierrez, C., Pérez, J.: Foundations of RDF Databases. In: Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M.-C., Schmidt, R.A. (eds.) Reasoning Web 2009. LNCS, vol. 5689, pp. 158–204. Springer, Heidelberg (2009)
3. Lausen, G.: Relational Databases in RDF: Keys and Foreign Keys. In: Christophides, V., Collard, M., Gutierrez, C. (eds.) SWDB-ODBIS 2007. LNCS, vol. 5005, pp. 43–56. Springer, Heidelberg (2008)
4. Lausen, G., Meier, M., Schmidt, M.: SPARQLng constraints for RDF. In: EDBT, pp. 499–509 (2008)
5. Martín, L., Anguita, A., Jiménez, A., Crespo, J.: Enabling cross constraint satisfaction in RDF-based heterogeneous database integration. In: ICTAI (2), pp. 341–348 (2008)
6. RDF vocabulary description language 1.0: RDF schema (2004), http://www.w3.org/TR/rdf-schema/
7. RDF current status, http://www.w3.org/standards/techs/rdf#w3c_all
8. RDF primer (2004), http://www.w3.org/TR/rdf-primer/
9. RDF semantics (2004), http://www.w3.org/TR/rdf-mt/
10. Sparql protocol and RDF query language (sparql) (2008), http://www.w3.org/TR/rdf-sparql-query/
11. Thalheim, B.: A compelte axiomatization for full join dependencies in relations. Bulletin of the EATCS 24, 109–114 (1984)

Some Remarks on Relational Database Schemes Having Few Minimal Keys

Joachim Biskup

Technische Universität Dortmund, Dortmund, Germany
joachim.biskup@cs.tu-dortmund.de

Abstract. Relational database schemes comprise semantic constraints to formally capture at least part of the "real-world" semantics of an application. Functional dependencies constitute a basic and widely studied class of such constraints. Accordingly, many properties of this class are known, including the insight that the number of minimal keys – as determined by a declared set of functional dependencies – might vary extensively, from just one to exponentially many (in the number of the underlying attributes). The case of just one minimal key is known to be characterized by the set of extremal attributes forming a minimal key. Starting from this result, the present work studies schemes having only a few minimal keys. In particular, we consider the cases of schemes having two and three minimal keys, and then suggest some research for dealing with the more general case of n minimal keys. Furthermore we study the impact of additionally requiring the schemes to be in Boyce-Codd normal form or Third normal form.

Keywords: Boyce-Codd normal form, computational complexity, extremal attribute, functional dependency, functional relationship, implicational closure, logical implication, minimal key, minimal-key equivalence, NP-completeness, object normal form, prime attribute, relational database, relation scheme, semantic modeling, Sperner system, super-prime attribute, Third normal form.

1 Introduction

Database technology enables a system administrator to formally express "real-world" semantics of the pertinent application by using a Data Definition Language, DDL. Such a language might be very expressive to deal with advanced concepts of *semantic modeling*, as, e.g., originally suggested for the ER-model by Chen [7] and later on further elaborated by Thalheim [20], who recently also provided a critical review of the goals and the achievements of semantic modeling [21], and by many others as well. The data definition language usually features means to denote recognized "real-world" aspects by attributes, to group attributes together, and then, for each resulting group, to specify *functional relationships* as a simple kind of semantic constraints. A functional relationship roughly requires that (the values of) a subset of grouped attributes uniquely determines (the values of) another subset.

In the setting of *relational databases*, applying these means results in forming one or more relation schemes with *functional dependencies*. Though, evidently, a relation scheme having only functional dependencies as declared semantic constraints only captures rather simple and thus restricted "real-world" semantics, intensive research has indicated that the class of such schemes has a rich and nontrivial structure raising challenging research problems, see, e.g., the compendium by Thalheim [18], the monographs by Paredaens et al [16], Mannila/Räihä [15] and Atzeni/De Antonellis [2], or the textbook summary by Abiteboul et al [1].

One direction of this research aims to study the set of *minimal keys* of a relation scheme with functional dependencies and, in particular, the number of different minimal keys, see the (temporally sorted) contributions [14,8,9,18,19,10,4] of the (alphabetically sorted) authors Biskup, Demetrovics, Katona, Libkin, Lucchesi, Muchnik, Miklos, Osborn, Seleznjev and Thalheim, as well as many further investigations. It fact, this number might vary extensively, from just one to exponentially many (in the number of the underlying attributes).

These "Remarks on ... Few Minimal Keys" focus on relation schemes having only a few minimal keys. Though restricting to relational databases, the presented insight might also have an impact on more expressive data models, all of which comprise a suitably adapted notion of a minimal key while, e.g., admitting null values [17], more generally allowing incomplete data [11], dealing with description logic (rather than first-order logic) [22], treating semi-structured XML-data [6,12], or referring to RDF-items [13]. In fact, the intuitive notion of a (minimal) key is a fundamental construct of any approach to semantic modeling in general and the various formalized data models in particular.

2 Basic Notations and Results

In this section we briefly introduce the basic notation for relation schemes, functional dependencies and minimal keys, see, e.g., the volumes [1,2,15,16,18] already cited above, and we summarize the fundamental results about schemes having a unique minimal key, see [4].

We consider a single *(relation) scheme* $RS = \langle U, \Sigma \rangle$, where the universe U is a finite set of *attributes* with $|U| = n$, and Σ is a finite set of *semantic constraints* on U, which we assume to be functional dependencies – the most prevalent kind of local constraints in form of data dependencies in actual relational databases.

An *instance* r of a relation scheme is a finite set of tuples over the universe U satisfying Σ, in the sense of first-order logic considering an instance as a finite Herbrand interpretation. The values c_i of a *tuple* $\mu = (c_1, \ldots, c_n)$ are elements of an infinite set of constants, and the value of an attribute $a \in U$ in a tuple μ is referred to by $\mu[a]$. Similarly, the values of a set of attributes $X \subseteq U$ in a tuple μ are denoted by $\mu[X]$.

Let $X, Y \subseteq U$ be sets of attributes, then r satisfies the *functional dependency (FD)* $X \rightarrow Y$ if for any two tuples μ_1, μ_2 of r it holds that $\mu_1[Y] = \mu_2[Y]$ whenever $\mu_1[X] = \mu_2[X]$. $K \subseteq U$ is a *key* of RS (in some work called a "superkey") if $K \rightarrow U$ is logically implied by Σ. If additionally K is $\subseteq$-minimal with this property, it is a *minimal key* (in some work simply called a "key"). Obviously, any two different minimal keys are not contained in each other, and thus the set of all minimal keys forms a Sperner system, as first observed in [8,9].

A scheme RS is in *Boyce-Codd normal form (BCNF)* if for each FD $X \rightarrow Y$ logically implied by Σ and with $Y \not\subseteq X$, X is a key. If an attribute $a \in U$ is an element of some minimal key, it is called *prime*; otherwise, if it is not contained in any minimal key, it is called *nonprime*. In general strictly less restrictive than BCNF, a scheme RS is in *Third normal form (3NF)* if for each FD $X \rightarrow a$ logically implied by Σ and with a being a nonprime attribute[1] such that $a \notin X$, X is a key. As shown in [23], if a scheme is in 3NF but not in BCNF, then there exists a pair of overlapping minimal keys.

Logical implication among (sets of) functional dependencies is denoted by $\models$. If $\Pi \models X \rightarrow Y$, we also write $X \rightarrow Y \in \Pi^+$. To avoid trivial cases (where an instance may have at most one tuple), throughout these remarks we only consider schemes $RS = \langle U, \Sigma \rangle$ such that $\emptyset \rightarrow U \notin \Sigma^+$.

An implication of a functional dependency $X \rightarrow Y$ by a set of functional dependencies Π can be decided by computing the *closure* $\Pi[X]$ of the set of attributes X under the functional dependencies in Π, the *Π-closure* of X for short, and then checking whether $Y \subseteq \Pi[X]$:

$$\Pi \models X \rightarrow Y \quad \text{iff} \quad Y \subseteq \Pi[X] \,. \tag{1}$$

A set of attributes X is *closed* under a set of functional dependencies Π, if $\Pi[X] = X$.

Some of the concepts mentioned can be efficiently computed, but others are of high computational complexity. We can efficiently determine the *Π-closure* of X by exhaustively using the FDs in Π as conditional production rules: the output item is initialized with the argument X, and then an FD in Π of the form $R \rightarrow S$ adds the attributes in S to the output item under the condition that R has been a subset of the output item before. Accordingly, basically by (1), implications of FDs and the minimal key property of attribute sets can be treated efficiently as well. Moreover, somehow surprisingly, BCNF for a full scheme $\langle U, \Sigma \rangle$ can be tested efficiently too, since it suffices to only consider the elements in Σ (rather than all those that are in Σ^+).

In contrast, we are faced with the possibility that a scheme might have exponentially many minimal keys. Moreover, to test the primality of an attribute we might be forced to consider essentially all of the potentially many minimal keys. In fact, the primality property is NP-complete, where a nondeterministic affirmative test just guesses a suitable minimal key and then efficiently confirms the required properties. Furthermore, the problem whether a scheme admits a minimal key of cardinality less than m is NP-complete as well. Finally – in contrast to the efficiency result stated above – BCNF-testing for a subscheme defined by a strict subset $X \not\subseteq U$ of the universe U requires us to project the given FDs on X potentially leading to substantial computational costs. Note that such tests are

[1] Following common conventions, we will omit set brackets for singleton sets and occasionally use other abbreviations for operations on sets of attributes.

part of the well-known decomposition algorithm to stepwise transform a given relation scheme into a collection of subschemes each of which being in BCNF.

3 Schemes Having One Minimal Key

As recalled above, a relation scheme can possess a highly complex structure, though the witnessing examples are mainly obtained by purely theoretical constructions that will rarely reflect the semantic model of some practical application. However, while modeling and formalizing "real-world" aspects of an advanced and comprehensive application, an administrator might well be faced with both application-intrinsic complexities like overlapping minimal keys or cyclic functional relationships and model-specific challenges resulting from a limited expressiveness of the available formal means. Clearly, as far as possible, an administrator will aim at avoiding to declare schemes that exhibit an unnecessary degree of complexity.

In the simplest and best manageable case, there exists just one minimal key for each scheme considered. This case can be efficiently recognized and treated as described in the remainder of this section.

Given a scheme $\langle U, \Sigma \rangle$, an attribute $a \in U$ is called *extremal* if $U \setminus a \to a \notin \Sigma^+$, i.e., $U \setminus a$ is closed under Σ. The set E of all extremal attributes is contained in each minimal key; conversely, if an attribute a is *superprime* in the sense that it belongs to all the minimal keys, then it is extremal (Lemma 1 and Cor. 1 of [4]):

$$a \text{ is extremal} \quad \text{iff} \quad a \text{ is superprime.} \tag{2}$$

Moreover, a scheme has exactly one minimal key if and only if the set of extremal attributes forms a (minimal) key (Thm. 1 of [4]), i.e.:

$$\langle U, \Sigma \rangle \text{ has a unique minimal key} \quad \text{iff} \quad \{a \mid U \setminus a \to a \notin \Sigma^+\} \to U \in \Sigma^+. \tag{3}$$

We can characterize such schemes in an alternative way by considering the attributes that do not trivially appear in the right hand side of any FD in Σ (Thm. 3.4 of [2]), i.e.,

$$\langle U, \Sigma \rangle \text{ has a unique minimal key} \quad \text{iff} \quad (U \setminus \bigcup_{X \to Y \in \Sigma} Y \setminus X) \to U \in \Sigma^+. \tag{4}$$

Finally, consider a scheme $\langle U, \Sigma \rangle$ with Σ being a minimal cover – i.e., containing only elementary FDs (with singleton right-hand sides) and being redundancy-free both element-wise (with minimal left-hand sides) and globally (without an FD implied by the remaining ones) – and the set E of its extremal attributes being the unique minimal key. Then this scheme is in BCNF (actually the scheme then is in *Object normal form* [3]) if and only if $\Sigma = \{E \to b \mid b \in U \setminus E\}$ (Thm. 3 of [4]).

4 Schemes Having Two Minimal Keys

As recalled in Section 3, a scheme $\langle U, \Sigma \rangle$ with $\Sigma[E] = U$ has *exactly one* minimal key, namely E. We will now start considering the case that $\Sigma[E] \neq U$, which implies that there are *at least two* minimal keys. Let us assume that $E \neq \emptyset$, and we define $D := \Sigma[E] \setminus E$ to be the set of *dependent* attributes, and $I := U \setminus ED \neq \emptyset$ the set of *independent* attributes.

We first observe that each attribute $a \in D$ is *nonprime*, i.e., it cannot be contained in any minimal key (since, if it was an element of a minimal key K, then $E \subseteq K$ by (2) and thus a could be dropped from K without affecting K being a key; thus the minimality of K would be violated). Furthermore, I must contain at least two elements (since otherwise, the single element $a \in I$ would be extremal; thus $E \cap I = \emptyset$ would be violated).

So let us assume that $I = \{a, b\}$ for $a \neq b$. Then the proposition presented below states that – under the assumptions made so far – there are exactly two minimal keys. Afterwards we argue that – essentially – the converse statement also holds.

Proposition 1. *Let $RS = \langle U, \Sigma \rangle$ be a scheme such that $E \neq \emptyset$, $\Sigma[E] \neq U$, and $I = \{a, b\}$ for $a \neq b$. Then RS has exactly two minimal keys, namely $E \cup a$ and $E \cup b$.*

Proof. According to the assumptions, E is not a minimal key by (3), and thus, by (2), a minimal key has the form $E \cup a$, $E \cup b$ or $E \cup ab$. Considering these options in turn, we will show that the third option is not possible, and that the remaining two options only occur together.

Case 1, $E \cup a$ is a minimal key: Then $E \cup ab$ is a key but not minimal. Furthermore, assume indirectly that $E \cup b$ is not a minimal key. Then $E \cup b \to a \notin \Sigma^+$, and thus also $E \cup D \cup b \to a \notin \Sigma^+$, where we have $E \cup D \cup b = U \setminus a$. Thus a is extremal, a contradiction to $a \notin E$.

Case 2, $E \cup b$ is a minimal key: Analogously.

Case 3, $E \cup a$ is not a minimal key and $E \cup b$ is not a minimal key: Then $E \cup ab$ is the unique minimal key, and thus both attributes a and b are extremal by (2), a contradiction to $a, b \notin E$. □

Conversely, consider a scheme $\langle U, \Sigma \rangle$ that has exactly two minimal keys $K_1 \neq K_2$. Then $E = K_1 \cap K_2$ by (2), and $E \cup (K_1 \setminus K_2) \to U \in \Sigma^+$ and $E \cup (K_2 \setminus K_1) \to U \in \Sigma^+$. Comparing this situation with that of Proposition 1, we see that the nonempty set $K_1 \setminus E$ corresponds to the single attribute a, and the nonempty set $K_2 \setminus E$ to the single attribute b.

To capture these correspondences more precisely, for a given scheme RS we define that attributes a and b of the universe are $(RS\text{-})$*minimal-key equivalent*, $a \sim_{mk} b$, if for all minimal keys K of the scheme we have that $a \in K$ iff $b \in K$. Obviously, $\sim_{mk}$ is an equivalence relation over U, where, in particular, the set E of all extremal attributes forms an equivalence class, and so does the set N of all nonprime attributes, provided these sets are nonempty. Also note that the relation $\sim_{mk}$ – and thus its corresponding set $\{[a]_{\sim_{mk}} \mid a \in U\}$ of equivalence classes $[a]_{\sim_{mk}}$ – reflects the intersection structure of the set of all minimal keys. Accordingly, we might also consider the notion of an attribute being *l-prime*, meaning that it is an element of exactly l minimal keys.

Minimal-key equivalence seems to be of high computational complexity in general. However, if the scheme is in BCNF and specified by a minimal cover (which can always be found efficiently), then the set of all minimal keys coincides with the set of all left-hand sides of functional dependencies occurring in the minimal cover. For, if L is a left-hand side, by the BCNF property, L must contain a minimal key $K \subseteq L$, where $K \neq L$ would violate the minimal cover condition; conversely, if K is a minimal key, then $K \neq U$, since $\Sigma[E] \neq U$, and thus there is an FD $L \rightarrow a$ in the minimal cover such that $L \subseteq K$, in fact by the BCNF property we have $L = K$. Accordingly, then $a \sim_{mk} b$ can be easily decided by checking whether for all left-hand sides L occurring in the minimal cover we have that $a \in L$ iff $b \in L$.

Applying the notions introduced above to the comparison of the special situation of Proposition 1 with the more general case, we see the following. Each of the attributes a and b of the special situation is 1-prime and forms a singleton $\sim_{mk}$-equivalence class; correspondingly, in the general situation each of the sets $K_1 \setminus E$ and $K_2 \setminus E$ consists of 1-prime attributes and forms an $\sim_{mk}$-equivalence class.

Summarizing and formalizing the arguments given so far, the general situation of schemes having exactly two minimal keys is fully characterized by the following Theorem 1 and visualized in Figure 1.

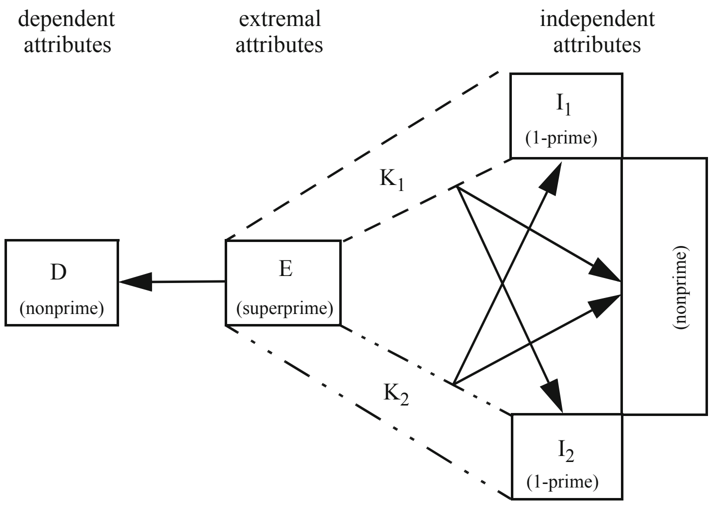


Fig. 1. Dependency structure of schemes having exactly two minimal keys

Theorem 1. *Let $RS = \langle U, \Sigma \rangle$ be a scheme. Then RS has exactly two minimal keys $K_1 \neq K_2$ iff (1) $\Sigma[E] \neq U$, and (2) $\sim_{mk}$ has exactly two equivalence classes I_1 and I_2 that are different from E and N, and (3) $K_1 = E \dot{\cup} I_1$ and $K_2 = E \dot{\cup} I_2$, and thus then $I_1 = K_1 \setminus K_2$ and $I_2 = K_2 \setminus K_1$.*

Proof. Immediate from the consideration presented before the theorem. Note that property (1) is redundant since it follows from property (2). □

Finally, we indicate requirements for a scheme with exactly two different keys $K_1 \neq K_2$ being in BCNF. If either $E = \emptyset$ or both $E \neq \emptyset$ and $D = \emptyset$, then BCNF is achievable by $\Sigma = \{K_1 \to b \mid b \in U \setminus K_1\} \cup \{K_2 \to a \mid a \in U \setminus K_2\}$, which is a minimal cover. However, in the remaining case, i.e., $E \neq \emptyset$ and $D \neq \emptyset$, BCNF is not achievable, since the FD $E \to D$ would be implied thereby violating BCNF.

By the same reason, that remaining case is not compatible with 3NF either. Furthermore we observe the following: if the scheme is in 3NF but not in BCNF, then K_1 and K_2 are overlapping by the result of [23], and thus we have both $E \neq \emptyset$ and $D = \emptyset$; the scheme $\langle \{e, a, b\}, \{ea \to b, b \to a\} \rangle$ with keys ea and eb is a well-known example of this situation.

5 Schemes Having Three Minimal Keys

Let us now assume that I has exactly three elements, say $I = \{a, b, c\}$. If we do not have the situation described by Theorem 1, then the proposition presented below states that there are exactly three minimal keys. Afterwards we again argue that – essentially – the converse statement also holds.

Proposition 2. *Let $RS = \langle U, \Sigma \rangle$ be a scheme such that $E \neq \emptyset$, $\Sigma[E] \neq U$, RS has more than two minimal keys, and $I = \{a, b, c\}$ has cardinality 3. Then RS has exactly three minimal keys, namely either $E \cup a$, $E \cup b$ and $E \cup b$ or $E \cup ab$, $E \cup bc$ and $E \cup ca$.*

Proof. According to the assumptions, E is not a minimal key by (3), and furthermore, similarly to the argument of Case 3 of the proof of Proposition 1, $E \cup abc$ cannot be a minimal key, since then it would be unique and thus we would have $a, b, c \in E$. Thus, by (2), a minimal key has the form $E \cup a$, $E \cup b$, $E \cup ab$, $E \cup ab$, $E \cup bc$ or $E \cup ca$.

Since minimal keys may not include each other, besides the cases mentioned in the proposition only combinations of two keys of the form $E \cup x$ and $E \cup yz$ are possible, which however would violate the assumption that there are more than two minimal keys. □

Note that in both cases mentioned in Proposition 2, each of the attributes a, b and c forms a singleton $\sim_{mk}$-equivalence class, each of these attributes being 1-prime in the first case, but 2-prime in the second case. Conversely, consider now a scheme $\langle U, \Sigma \rangle$ that has exactly three minimal keys K_1, K_2 and K_3. Then $E = K_1 \cap K_2 \cap K_2$ by (2), and the characterization stated in Theorem 1 does not hold. Comparing this situation with that of Proposition 2, we see that the

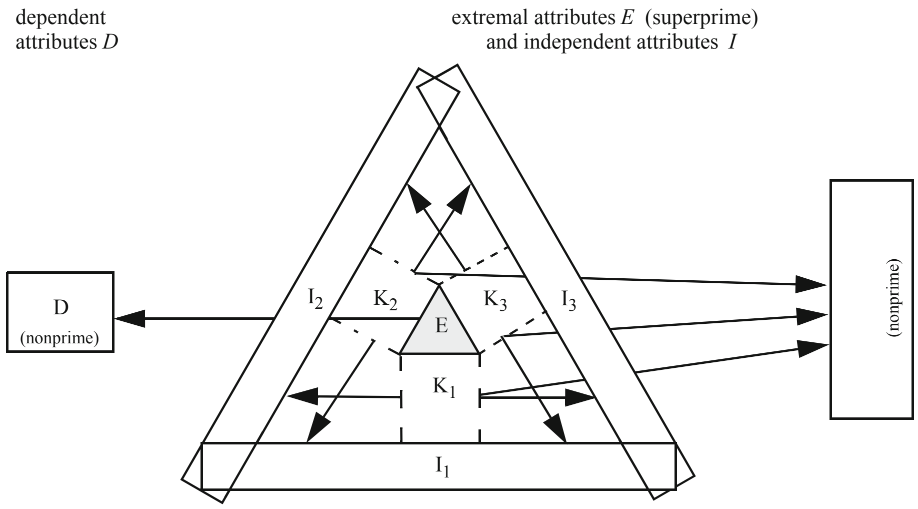


Fig. 2. Dependency structure of schemes having exactly three minimal keys

minimal keys K_1, K_2, K_3 corresponds to either $\{a\}$, $\{b\}$, $\{c\}$ or $\{a, b\}$, $\{b, c\}$, $\{c, a\}$. As in Section 4, a single attribute in Proposition 2 might have a set of equally treated (i.e., $\sim_{mk}$-equivalent) attributes as a counterpart. Moreover, without the restriction of having exactly three elements in I, besides the cases of having either no (nonempty) intersections at all or all possible intersections, the remaining cases might appear as well.

Summarizing, the general situation of schemes having exactly three minimal keys is fully characterized by the following Theorem 2 – where again property (1) is redundant since it follows from property (2) – and visualized in Figure 2.

Theorem 2. *Let $RS = \langle U, \Sigma \rangle$ be a scheme. Then RS has exactly three minimal keys K_1, K_2 and K_3 iff (1) $\Sigma[E] \neq U$, and (2) $\sim_{mk}$ has at least three and at most six equivalence classes that are different from E and N, where (3) of those classes at most three consist of 1-prime attributes, at most three consist of 2-prime attributes, and none consists of l-prime attributes with $l > 3$.*

Proof. The "only-if part " is easily verified by inspecting Figure 2. The "if-part" is justified by distinguishing the four main cases that there are j equivalence classes, for $j = 3, 4, 5, 6$, and for each of them the subcases according to the contribution of the classes with 1-prime and 2-prime elements, respectively. For $j = 3$, we recognize the situations described in Proposition 2. For $j = 6$, the situation as shown in Figure 2 arises, assuming that each set I_i has "private" (1-prime) attributes and nonempty intersections with the other I_js. For $j = 4$ and $j = 5$, some of the "private" parts are missing or some of the intersections are empty: in all cases however, we can form exactly three different I_1, I_2 and I_3 that do not contain one another, and thus each of them together with E is a minimal key. □

Finally, like in the preceding sections, to achieve BCNF while having exactly three minimal keys, a minimal cover Σ must have exactly three left-hand sides L_1, L_2 and L_3, none of them containing another, and each of them serving as a minimal key, i.e., $\Sigma = \bigcup_{i=1,2,3}\{L_i \rightarrow x \mid x \in U \setminus L_i\}$.

Again, as with two minimal keys, the case that both $E \neq \emptyset$ and $D \neq \emptyset$ is excluded for 3NF, and thus for BCNF as well. Moreover, if the scheme is in 3NF but not in BCNF, at least two of the three keys must have a nonempty intersection, by the result of [23].

6 Conclusion

Our remarks on few minimal keys are not too surprising: sets of minimal keys and Sperner systems correspond to each other, and thus there are only a few options to construct them if the underlying set is small or the cardinality of the system is small. Somehow astonishing, however, is the observation that our characterizations of schemes having two or three minimal keys, respectively, appear to be a bit clumsy. In fact, we would like to employ a simple criterion for deciding whether an attribute a appears in exactly l minimal keys, in particular for a small value of l. Similarly, we would like to characterize minimal-key equivalence in a simple way. Such characterizations could then be employed to describe schemes having exactly n minimal keys, for $n > 3$. There seems to be no explicit studies on such problems so far, but we expect that they are not efficiently computable, given the complexity results on key-related problems cited before.

Besides of interest to an administrator charged to declare a "good" database scheme capturing a specification that results form semantic modeling, the situations analyzed in these remarks might also be relevant for a security officer attempting to protect confidential information from being inferable from query answers and a publicly known schema declaration, as exemplarily studied in [5].

Acknowledgments. I would like to sincerely thank both an anonymous reviewer for highly constructive remarks and suggestions, which led to a substantial improvement of the presentation, and Ralf Menzel for carefully reading and commenting the manuscript. Moreover, I am gratefully indebted to Bernhard Thalheim: both his great scientific achievements about semantic modeling and database theory and a most fruitful and challenging personal tradition of discussing these and other topics with him as a colleague and friend – in particular spanning the conference series MFDBS, ICDT and FoIKS – inspired not only this article but many further work of mine as well.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. Atzeni, P., Antonellis, V.D.: Relational Database Theory. Benjamin/Cummings, Redwood City (1993)

3. Biskup, J.: Boyce-Codd normal form and object normal forms. Inf. Process. Lett. 32(1), 29–33 (1989)
4. Biskup, J., Demetrovics, J., Libkin, L., Muchnik, I.B.: On relational database schemes having unique minimal key. Elektronische Informationsverarbeitung und Kybernetik 27(4), 217–225 (1991)
5. Biskup, J., Embley, D.W., Lochner, J.-H.: Reducing inference control to access control for normalized database schemas. Inf. Process. Lett. 106(1), 8–12 (2008)
6. Buneman, P., Davidson, S.B., Fan, W., Hara, C.S., Tan, W.C.: Reasoning about keys for XML. Inf. Syst. 28(8), 1037–1063 (2003)
7. Chen, P.P.: The entity-relationship model – toward a unified view of data. ACM Trans. Database Syst. 1(1), 9–36 (1976)
8. Demetrovics, J.: On the number of candidate keys. Inf. Process. Lett. 7(6), 266–269 (1978)
9. Demetrovics, J.: On the equivalence of candidate keys with Sperner systems. Acta Cybern. 4, 247–252 (1980)
10. Demetrovics, J., Katona, G.O.H., Miklós, D., Seleznjev, O., Thalheim, B.: Asymptotic properties of keys and functional dependencies in random databases. Theor. Comput. Sci. 190(2), 151–166 (1998)
11. Hartmann, S., Leck, U., Link, S.: On Codd families of keys over incomplete relations. Comput. J. 54(7), 1166–1180 (2011)
12. Hartmann, S., Link, S.: Efficient reasoning about a robust XML key fragment. ACM Trans. Database Syst. 34(2) (2009)
13. Lausen, G., Meier, M., Schmidt, M.: SPARQLing constraints for RDF. In: Kemper, A., et al. (eds.) International Conference on Extending Database Technology, EDBT 2008. ACM International Conference Proceeding Series, vol. 261, pp. 499–509. ACM, New York (2008)
14. Lucchesi, C.L., Osborn, S.L.: Candidate keys for relations. J. Comput. Syst. Sci. 17(2), 270–279 (1978)
15. Mannila, H., Räihä, K.-J.: The Design of Relational Databases. Addison-Wesley, Wokingham (1992)
16. Paredaens, J., Bra, P.D., Gyssens, M., Gucht, D.V.: The Structure of the Relational Database Model. Springer, Heidelberg (1989)
17. Thalheim, B.: On semantic issues connected with keys in relational databases permitting null values. Elektronische Informationsverarbeitung und Kybernetik 25(1/2), 11–20 (1989)
18. Thalheim, B.: Dependencies in Relational Databases. Teubner, Stuttgart (1991)
19. Thalheim, B.: The number of keys in relational and nested relational databases. Discrete Applied Mathematics 40(2), 265–282 (1992)
20. Thalheim, B.: Entity-Relationship Modeling – Foundations of Database Technology. Springer, Heidelberg (2000)
21. Thalheim, B.: Towards a Theory of Conceptual Modelling. In: Heuser, C.A., Pernul, G. (eds.) ER 2009. LNCS, vol. 5833, pp. 45–54. Springer, Heidelberg (2009)
22. Toman, D., Weddell, G.E.: On keys and functional dependencies as first-class citizens in description logics. J. Autom. Reasoning 40(2-3), 117–132 (2008)
23. Vincent, M.W., Srinivasan, B.: A note on relation schemes which are in 3NF but not in BCNF. Inf. Process. Lett. 48(6), 281–283 (1993)

Random Databases with Correlated Data

Gyula O.H. Katona

Rényi Institute, Budapest, Hungary
ohkatona@renyi.hu

Dedicated to Professor Bernhard Thalheim for his 60th birthday

Abstract. A model of random databases is given, with arbitrary correlations among the data of one individual. This is given by a joint distribution function. The individuals are chosen independently, their number m is considered to be (approximately) known. The probability of the event that a given functional dependency $A \to b$ holds (A is a set of attributes, b is an attribute) is determined in a limiting sense. This probability is small if m is much larger than $2^{H_2(A\to b)/2}$ and is large if m is much smaller than $2^{H_2(A\to b)/2}$ where $H_2(A \to b)$ is an entropy like functional of the probability distribution of the data.

1 Introduction

Consider the data of a class in a school (in Europe), it can be supposed that the last name is a key, that is all other data are functionally dependent on it. Considering the whole school, the probability of having two students with the same last name is pretty high, so the last name cannot be taken as a key. But, very likely the first and last names together from a key. It will be certainly not true for the data of a large city.

The example above illustrates that, considering the database to be random, the size (number of rows) largely determines which functional dependencies can be considered valid. The aim of the present paper is to give a model of this situation. The first attempts in this directions were the papers of Demetrovics, Katona, Miklós, Seleznjev and Thalheim [1], [2]. There the authors supposed that the data of one individual are probabilistically independent. It was shown even in this case that a set of *constant times the logarithm of the size of the database* many columns will functionally determine a given other column with high probability. Their model however was not able to include "real" functional dependencies or situations like "very probably functionally dependent". The aim of the present paper is to extend the results in this direction.

Let Ω be the set of attributes, $|\Omega| = n$. The set of all possible entries is denoted by E. (If the distinct attributes have different sets of entries then E is their union.) Let one row of the database is the random vector $(\xi_1, \xi_2, \ldots, \xi_n)$ where the ξs are not necessarily independent, the distribution is given by the probabilities

$$\Pr(\xi_1 = u_1, \xi_2 = u_2, \ldots, \xi_n = u_n) \tag{1}$$

for all possible entries $u_1, u_2, \ldots, u_n \in E$. Let $A \subset \Omega, b \in \Omega$. We say that b *functionally depends on A with probability one* if the probabilities

$$\Pr(\xi_i = u_i (i \in A), \xi_b = u_b)$$

are zero for all but one $u_b \in E$ for any choice of entries $u_i \in E (i \in A)$. On the other hand the individuals, the rows are chosen independently. In terms of probability theory, consider m (totally) independently chosen realizations of the random vector whose probabilities are determined by (1).

An entropy like function is needed to our further investigations. Let ξ and η be two, not necessarily independent random variables. The probability of the event that $\xi = k$ and $\eta = \ell$ is $p_{k,\ell}$, the probability of ξ being k is $p_k = \sum_\ell p_{k\ell}$. Define

$$H_2(\xi \to \eta) = -\log_2 \left(\sum_k p_k^2 - \sum_{k,\ell} p_{k,\ell}^2 \right). \tag{2}$$

This quantity is related to the Rényi entropy of order 2 (see [4] and [5]).

Let $A \subset \Omega, b \in \Omega, b \notin A$. The random vector of the coordinates $\xi_i (i \in A)$ will be denoted by α. The probability of the event that α is equal to the kth sequence is denoted by $p_k(A)$. Moreover, the probability of the event that α is equal to the kth sequence and ξ_b has the ℓth entry is $p_{k,\ell}(A, b)$. Our crucial notion is defined in the following way:

$$H_2(A \to b) = H_2(\alpha \to \xi_b). \tag{3}$$

The Heuristic Version of the Theorem. *The functional dependency $A \to b$ "seems to hold" (there are no two rows equal in the entries belonging to A and different in the column of b) with large probability in a random database of size m if and only if $2\log_2 m$ is much smaller than $H_2(A \to b)$.*

The statement above will be made more clear by analyzing two special cases. First let us suppose that all the ξ's are independent in (1) and each of them has a probability distribution $(q_1, q_2, \ldots, q_R)$. Then the probabilities in question are

$$\Pr(\xi_i = u_i (i \in A), \xi_b = u_b) = \prod_{i \in A} q_{u_i} \cdot q_{u_b}.$$

These probabilities will pay the role of $p_{k\ell}$ in (2), while p_k will be

$$\Pr(\xi_i = u_i (i \in A)) = \prod_{i \in A} q_{u_i}.$$

It is easy to see that in this case (deleting the arguments A and b)

$$\sum_k p_k^2 - \sum_{k,\ell} p_{k\ell}^2 = \sum_k p_k^2 - \sum_k p_k^2 \sum_\ell r_\ell^2 = \left(\sum_k p_k^2 \right) - \left(1 - \sum_\ell r_\ell^2 \right). \tag{4}$$

On the other hand,

$$\sum_\ell r_\ell^2 = \left(\sum_i q_i^2 \right)^{|A|}. \tag{5}$$

Using (2), (3), (4) and (5) we obtain

$$H_2(A \to b) = |A| \cdot H_2(q_1, q_2, \dots q_R) - \log_2 \left(1 - \sum_{\ell} q_\ell^2\right)$$

where $H_2(q_1, q_2, \dots q_R) = \log_2 \sum_i q_i^2$ is the Rényi entropy of order 2 ([4], [5]). Here the first term tends to infinity with $|A|$ while the second term is constant. $H_2(A \to b)$ is close to $|A| \cdot H_2(q_1, q_2, \dots q_R)$. The Theorem means in this case that $A \to b$ holds for a random database of size m if $2 \log_2 m$ is less than $|A| \cdot H_2(q_1, q_2, \dots q_R)$, that is, for the As satisfying $|A| > \frac{2 \log_2 m}{H_2(q_1, q_2, \dots q_R)}$. This was proved in [2].

The other important special case is when b is really functionally dependent on A. Then $p_{k\ell} = p_k$ for a uniquely determined $\ell = \ell(k)$, all other $p_{k\ell}$s are zero. Therefore the last term in (2) is equal to $\sum_k p_k^2$, (2) is plus infinity. The Theorem says in this case that $A \to b$ holds when $2 \log_2 m$ is less than ∞, that is always.

2 The Exact Forms of the Theorem

It will be supposed that the database consists of m (totally) independently chosen rows of the random vector defined by the probability distribution (1). Our result is of asymptotic nature, it is valid for large matrices, large number of columns and rows. More precisely we will assume that $n = |\Omega|, |A|$ depend on m what tends to infinity. It may seem more natural to take n to be the main variable and to suppose that the other quantities depend on it while it tends to infinity. However the size of the asymptotical existence of $A \to b$ is independent on n it only depends on the relation of m and $H_2(A \to b)$. This is why it is better to consider m as the basic variable.

It will be supposed that the distribution (1) for n' is the "continuation" of the one for n, that is, the probabilities in (1) can be obtained by summing the probabilities for n' for $\xi_{n+1}, \dots, \xi_{n'}$. The column b is fixed, while $|A|$ tends to infinity by adding newer and newer columns (distinct from b) to A.

Some more definitions are needed to the formulation of the theorem. The probability of the of the event that $A \to b$ $(A \subset \Omega, b \in \Omega)$ holds in a database of size m is denoted by $\Pr(A \to b, m)$. Let $p(\alpha, \xi_b, I)$ denote the probability of the event that the pair of two independent copies $(\alpha_1, \xi_{b,1}), (\alpha_2, \xi_{b,2})$ gives a counter-example, that is, $\Pr(\alpha_1 = \alpha_2, \xi_{b,1} \neq \xi_{b,2})$. Similarly $p(\alpha, \xi_b, V)$ denotes the probability of the event that the triple $(\alpha_1, \xi_{b,1}), (\alpha_2, \xi_{b,2}), (\alpha_3, \xi_{b,3})$ gives two counter-examples in the following way: $\alpha_1 = \alpha_2 = \alpha_3, \xi_{b,1} \neq \xi_{b,2} \neq \xi_{b,3}$. Finally $p(\alpha, \xi_b, N)$ is the probability of the event that the quadruple $(\alpha_1, \xi_{b,1}), (\alpha_2, \xi_{b,2}), (\alpha_3, \xi_{b,3}), (\alpha_4, \xi_{b,4})$ gives three counter-examples forming a path: $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4, \xi_{b,1} \neq \xi_{b,2} \neq \xi_{b,3} \neq \xi_{b,4}$.

The first exact form of the Theorem is a repetition/implementation of the main theorem in [3]. This theorem is stated for two random variables. The only novelty here is that one of these variables is a random vector α. But this causes no real change. Therefore the theorem below needs no proof here. The interested reader is referred to [3].

Theorem 1

$$\Pr(A \to b, m) \to \begin{cases} 0 & \text{if } 2\log_2 m - H_2(A \to b) \to +\infty, \\ e^{-2^{a-1}} & \text{if } 2\log_2 m - H_2(A \to b) \to a, \\ 1 & \text{if } 2\log_2 m - H_2(A \to b) \to -\infty. \end{cases}$$

under the assumptions that

$$\frac{p(\alpha, \xi_b, V)^2}{p(\alpha, \xi_b, I)^3} \to 0 \tag{6}$$

and

$$\frac{p(\alpha, \xi_b, N)}{p(\alpha, \xi_b, I)^2} \to 0 \tag{7}$$

hold.

Although this is the most general form of the statement, known to us, it is difficult to check if the conditions (6) and (7) hold. However, exploiting the matrix structure in this case we can give weaker, but more natural conditions. Let $p_\kappa(A)$ denote the probability of the event that $\alpha = \kappa$. Moreover, $p_{\kappa,\ell}(A, b)$ denotes the probability of the event that $\alpha = \kappa, \xi_b = \ell$.

Theorem 2

$$\Pr(A \to b, m) \to \begin{cases} 0 & \text{if } 2\log_2 m - H_2(A \to b) \to +\infty, \\ e^{-2^{a-1}} & \text{if } 2\log_2 m - H_2(A \to b) \to a, \\ 1 & \text{if } 2\log_2 m - H_2(A \to b) \to -\infty. \end{cases}$$

under the following assumptions:

(i)

$$\frac{\max_\kappa p_\kappa(A)}{\sqrt{\sum_\kappa p_\kappa^2(A)}} \to 0, \tag{8}$$

(ii) There is a constant $0 < u < 1$ *independent of* A *such that*

$$\frac{\sum_{\kappa,\ell} p_{\kappa,\ell}^2(A, b)}{\sum_\kappa p_\kappa^2(A)} \leq u \tag{9}$$

hold.

Proof. We have to prove that (i) and (ii) imply both (6) and (7). Let us start with some elementary lemmas. The first two of them prove that if a sequence of non-negative numbers is given with a fixed sum and an upper bound c is given on them then the sum of their squares is maximized for a choice with (one exception) all members $= c$ or 0. We give the proof for sake of completeness.

Lemma 1. *Let the real numbers* $0 \leq a, b, c$ *satisfy the inequalities* $b \leq a \leq c \leq a + b$. *Then*

$$a^2 + b^2 \leq c^2 + (a + b - c)^2 \tag{10}$$

holds.

Proof. Consider the function $x^2 + (a + b - x)^2$. It is increasing from $\frac{a+b}{2}$. The conditions of the lemma imply (10), considering the values $x = a$ and $x = c$. □

Lemma 2. *Let $a_1, a_2, \ldots, a_N$ be non-negative real numbers with sum $\sum_i a = s$. If $a_i \leq c$ holds for all $1 \leq i \leq N$ then*

$$\sum_i a_i^2 \leq \left\lceil \frac{s}{c} \right\rceil c^2 \tag{11}$$

is true.

Proof. We use induction over N. The case $N = 1$ is trivial. Order the numbers in the following way: $a_1 \leq a_2 \leq \ldots \leq a_N \leq c$. If $a_N = c$ then delete this member and use the inductional hypothesis. Otherwise, if $a_N < c$ two cases will be distinguished. Firstly, if $c \leq a_{N-1} + a_N$ then apply Lemma 1 with $a = a_N, b = a_{N-1}$. Replacing a_N by c and a_{N_1} by $a_{N-1} + a_N - c$ a new sequence of numbers is obtained with the the same sum and non-decreased sum of squares. It is sufficient to prove the statement for this sequence, but this follows from the previous case, since it contains $a_N = c$. Secondly, if $c > a_{N-1} + a_N$ then apply Lemma 1 with $b = a_{N-1}, a = a_N, c = a_{N-1} + a_N$. The so obtained inequality, $a_{N-1}^2 + a_N^2 \leq (a_{N-1} + a_N)^2 + 0^2$ (what can be directly seen) shows that replacing a_{N-1} and a_N by $a_{N-1} + a_N$ and 0 the sum is unchanged, the sum of the squares is non-decreased. Since this sequence contains a 0, omitting this the induction can be used, again. □

Lemma 3. *Let $q_1, q_2, \ldots, q_N$ be non-negative real numbers, where all of these (including N) depend on n what tends to the infinity. Then*

$$\frac{\max_k q_k}{\sum_k q_k} \to 0 \tag{12}$$

implies

$$\frac{\sum_k q_k^2}{\left(\sum_k q_k\right)^2} \to 0. \tag{13}$$

Proof. Use (11) of Lemma 2 with $c(n) = \max_k q_k$ and $s = \sum_k q_k$:

$$\frac{\sum_k q_k^2}{\left(\sum_k q_k\right)^2} \leq \frac{\left(\frac{\sum_k q_k}{c(n)} + 1\right) c^2(n)}{\left(\sum_k q_k\right)^2} = \frac{c(n)}{\sum_k q_k} + \left(\frac{c(n)}{\sum_k q_k}\right)^2$$

shows that (12) really implies (13). □

Return to the proof of Theorem 2. Lemma 3 will be applied for the values $p_\kappa^2(A)$ in place of q_i. Condition (12) becomes exactly (8), therefore (13) gives

$$\frac{\sum_\kappa p_\kappa^4(A)}{\left(\sum_\kappa p_\kappa^2(A)\right)^2} \to 0. \tag{14}$$

Let us now give a lower estimate on $p(\alpha, \xi_b, I)$ using (ii) (that is (9))

$$p(\alpha, \xi_b, I) = \sum_{\kappa} p_{\kappa}^2(A) - \sum_{\kappa,\ell} p_{\kappa,\ell}^2(A, b) \geq (1-u) \sum_{\kappa} p_{\kappa}^2(A). \tag{15}$$

Recall that $p(\alpha, \xi_b, N)$ is the probability of the event that the quadruple $(\alpha_1, \xi_{b,1}), (\alpha_2, \xi_{b,2}), (\alpha_3, \xi_{b,3}), (\alpha_4, \xi_{b,4})$ gives three counter-examples forming a path: $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4, \xi_{b,1} \neq \xi_{b,2} \neq \xi_{b,3} \neq \xi_{b,4}$. This is a subevent of the event that $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4$. The probability of the latter one is $\sum_{\kappa} p_{\kappa}^4(A)$. Hence we have

$$p(\alpha, \xi_b, N) \leq \sum_{\kappa} p_{\kappa}^4(A). \tag{16}$$

(15) and (16) give an upper bound on the left hand side of (7):

$$\frac{p(\alpha, \xi_b, N)}{p(\alpha, \xi_b, I)^2} \leq \frac{\sum_{\kappa} p_{\kappa}^4(A)}{(1-u)^2 \left(\sum_{\kappa} p_{\kappa}^2(A)\right)^2}. \tag{17}$$

The right hand side tends to 0 by (14), proving (7).

The left hand side of (6) can be similarly upperbounded:

$$\frac{p(\alpha, \xi_b, V)^2}{p(\alpha, \xi_b, I)^3} \leq \frac{\left(\sum_{\kappa} p_{\kappa}^3(A)\right)^2}{(1-u)^3 \left(\sum_{\kappa} p_{\kappa}^2(A)\right)^3}. \tag{18}$$

Apply the well-known Cauchy-Bunyakovsky-Schwarz inequality

$$\left(\sum_i a_i b_i\right)^2 \leq \left(\sum_i a_i^2\right)\left(\sum_i b_i^2\right)$$

with $p_{\kappa}(A)$ and $p_{\kappa}^2(A)$:

$$\left(\sum_{\kappa} p_{\kappa}^3(A)\right)^2 \leq \left(\sum_{\kappa} p_{\kappa}^2(A)\right)\left(\sum_{\kappa} p_{\kappa}^4(A)\right).$$

This latter inequality implies

$$\frac{\left(\sum_{\kappa} p_{\kappa}^3(A)\right)^2}{\left(\sum_{\kappa} p_{\kappa}^2(A)\right)^3} \leq \frac{\sum_{\kappa} p_{\kappa}^4(A)}{\left(\sum_{\kappa} p_{\kappa}^2(A)\right)^2}. \tag{19}$$

(18), (19) and (14) prove (6). □

3 Remarks, Future Work

Related earlier work. In addition to the papers [1], [2], mentioned in the introduction, one should mention the important works of Seleznjev and Thalheim [6], [7] on the probabilistic-statistical properties of databases.

On the conditions of Theorem 2. Condition (i) is a rather weak one, it is satisfied in a very wide range. However it is stronger than the condition $\max_{\kappa} p_{\kappa}(A) \to 0$. An example when the latter one holds but (i) does not is the following. Let N denote the total number of members (probabilities), and choose the largest one to be $\frac{1}{\sqrt{N}}$, the other ones are equal, and add up to 1. Then the limit in (8) is $\frac{1}{2}$, not 0.

On the other hand, condition (ii) (that is (9)) is strong. It excludes *e.g.* the case when b is "very probably functionally dependent" on A. We are sure that (6) and (7) can be proved under (8) and a much weaker condition than (9). It needs a deeper analysis of the situation. For instance, the rough estimate (16) is not sufficient in this more general case.

Future work. Besides the analytic work suggested in the previous paragraph, one should consider a more general setting of the whole problem. Already the present setting has a certain "data mining" nature. We investigated here that when (at what size?) a hidden, weak statistical dependence becomes visible. A possible more general setting is when the following question is investigated. Given the statistical dependence a certain number of examples can be expected. At what size have we at least (say) half of this number. Another possible step forward if the "quality" of the examples is also taken into consideration.

References

1. Demetrovics, J., Katona, G.O.H., Miklós, D., Seleznjev, O., Thalheim, B.: Asymptotic properties of keys and functional dependencies in random databases. Theoretical Computer Sciences 190, 151–166 (1998)
2. Demetrovics, J., Katona, G.O.H., Miklós, D., Seleznjev, O., Thalheim, B.: Functional dependencies in random databases. Studia Sci. Math. Hungar. 34, 127–140 (1998)
3. Katona, G.O.H.: Testing functional connection between two random variables. Prokhorov Festschrift (accepted)
4. Rényi, A.: Some fundamental questions of information theory. MTA III Oszt. Közl. 10, 251–282 (1960) (in Hungarian)
5. Rényi, A.: On measures of information and entropy. In: Proc. of the 4th Berkeley Symposium on Mathematics, Statistics and Probability, pp. 547–561 (1960/1961)
6. Seleznjev, O., Thalheim, B.: Average Case Analysis in Database Problems. Methodology and Computing in Applied Probability 5(4), 395–418 (2003)
7. Seleznjev, O., Thalheim, B.: Random Databases with Approximate Record Matching. Methodology and Computing in Applied Probability 12(1), 63–89 (2010)

Statistical Inference for Rényi Entropy Functionals

David Källberg[1], Nikolaj Leonenko[2], and Oleg Seleznjev[1]

[1] Department of Mathematics and Mathematical Statistics,
Umeå University, SE-901 87 Umeå, Sweden
[2] School of Mathematics, Cardiff University,
Senghennydd Road, Cardiff CF24 4YH, UK

Abstract. Numerous entropy-type characteristics (functionals) generalizing Rényi entropy are widely used in mathematical statistics, physics, information theory, and signal processing for characterizing uncertainty in probability distributions and distribution identification problems. We consider estimators of some entropy (integral) functionals for discrete and continuous distributions based on the number of epsilon-close vector records in the corresponding independent and identically distributed samples from two distributions. The proposed estimators are generalized U-statistics. We show the asymptotic properties of these estimators (e.g., consistency and asymptotic normality). The results can be applied in various problems in computer science and mathematical statistics (e.g., approximate matching for random databases, record linkage, image matching).

AMS 2000 subject classification: 94A15, 62G20

Keywords: entropy estimation, Rényi entropy, U-statistics, approximate matching, asymptotic normality.

1 Introduction

Let X and Y be d-dimensional random vectors with discrete or continuous distributions $\mathcal{P}_X$ and $\mathcal{P}_Y$, respectively. In information theory and statistics, there are various generalizations of Shannon entropy (Shannon, 1948), characterizing uncertainty in $\mathcal{P}_X$ and $\mathcal{P}_Y$, for example, the Rényi entropy (Rényi, 1961, 1970),

$$h_s = h_s(\mathcal{P}_X) := \frac{1}{1-s}\log\left(\int_{R^d} p_X(x)^s dx\right), \qquad s \neq 1,$$

and the (differentiable) variability for approximate record matching in random databases

$$v = v(\mathcal{P}_X, \mathcal{P}_Y) := -\log\left(\int_{R^d} p_X(x)p_Y(x)dx\right),$$

where $p_X(x), p_Y(x), x \in R^d$, are densities of $\mathcal{P}_X$ and $\mathcal{P}_Y$, respectively (see Seleznjev and Thalheim, 2003, 2008). Henceforth we use $\log x$ to denote the natural logarithm of x. More generally, for non-negative integers $r_1, r_2 \geq 0$, $\mathbf{r} := (r_1, r_2)$, and $r := r_1 + r_2$, we consider *Rényi entropy functionals*

$$q_{\mathbf{r}} = q_{\mathbf{r}}(\mathcal{P}_X, \mathcal{P}_Y) := \int_{R^d} p_X(x)^{r_1} p_Y(x)^{r_2} dx,$$

and for the discrete case, $\mathcal{P}_X = \{p_X(k), k \in N^d\}$ and $\mathcal{P}_Y = \{p_Y(k), k \in N^d\}$,

$$q_{\mathbf{r}} = q_{\mathbf{r}}(\mathcal{P}_X, \mathcal{P}_Y) := \sum_k p_X(k)^{r_1} p_Y(k)^{r_2}.$$

Moreover, introduce the functionals

$$h_{\mathbf{r}} = h_{\mathbf{r}}(\mathcal{P}_X, \mathcal{P}_Y) := \frac{1}{1-r} \log q_{\mathbf{r}}, \quad r \neq 1.$$

Then, for example, the Rényi entropy $h_s = h_{s,0}$ and the variability $v = h_{1,1}$. Let $X_1, \ldots, X_{n_1}$ and $Y_1, \ldots, Y_{n_2}$ be mutually independent samples of independent and identically distributed (i.i.d.) observations from $\mathcal{P}_X$ and $\mathcal{P}_Y$, respectively. We consider the problem of estimating the entropy-type functionals $q_{\mathbf{r}}$, $h_{\mathbf{r}}$, and related characteristics for $\mathcal{P}_X$ and $\mathcal{P}_Y$ from the samples $X_1, \ldots, X_{n_1}$ and $Y_1, \ldots, Y_{n_2}$.

Various entropy applications in statistics (e.g., classification and distribution identification problems) and in computer science and bioinformatics (e.g., average case analysis for random databases, approximate pattern and image matching) are investigated in, e.g., Kapur (1989), Kapur and Kesavan (1992), Leonenko et al. (2008), Szpankowski (2001), Seleznjev and Thalheim (2003, 2008), Thalheim (2000), Baryshnikov et al. (2009), and Leonenko and Seleznjev (2010). Some average case analysis problems for random databases with entropy characteristics are investigated also in Demetrovics et al. (1995, 1998a, 1998b).

In our paper, we generalize the results and approach proposed in Leonenko and Seleznjev (2010), where inference for the quadratic Rényi entropy is studied for one sample. We consider properties (consistency and asymptotic normality) of kernel-type estimators based on the number of coincident (or ϵ-close) observations in d-dimensional samples for a more general class of entropy-type functionals. These results can be used, e.g., in evaluation of asymptotical confidence intervals for the corresponding Rényi entropy functionals.

Note that our estimators of entropy-type functionals are different form those considered by Kozachenko and Leonenko (1987), Tsybakov and van der Meulen (1996), Leonenko et al. (2008), and Baryshnikov et al. (2009) (see Leonenko and Seleznjev, 2010, for a discussion).

First we introduce some notation. Throughout the paper, let X and Y be independent random vectors in R^d with distributions $\mathcal{P}_X$ and $\mathcal{P}_Y$, respectively. For the discrete case, $\mathcal{P}_X = \{p_X(k), k \in N^d\}$ and $\mathcal{P}_Y = \{p_Y(k), k \in N^d\}$. In the continuous case, let the distributions be with densities $p_X(x)$ and $p_Y(x), x \in R^d$, respectively. Let $d(x, y) = |x - y|$ denote the Euclidean distance in R^d and $B_\epsilon(x) := \{y : d(x, y) \leq \epsilon\}$ an ϵ-ball in R^d with center at x, radius ϵ, and volume $b_\epsilon(d) = \epsilon^d b_1(d)$, $b_1(d) = 2\pi^{d/2}/(d\Gamma(d/2))$. Denote by $p_{X,\epsilon}(x)$ the ϵ-ball probability

$$p_{X,\epsilon}(x) := P\{X \in B_\epsilon(x)\}.$$

We write $I(C)$ for the indicator of an event C, and $|D|$ for the cardinality of a finite set D.

Next we define estimators of $q_\mathbf{r}$ when r_1 and r_2 are non-negative integers with $r_1 + r_2 \geq 2$. In order to include the one-sample case, we assume without loss of generality that $r_1 \geq r_2$. So, we have that $r_1 \geq 1$. Denote $\mathbf{n} := (n_1, n_2)$, $n := n_1 + n_2$, and say that $\mathbf{n} \to \infty$ if $n_1, n_2 \to \infty$. For $r_2 \neq 0$, let $p_\mathbf{n} := n_1/n \to p, 0 < p < 1$, as $\mathbf{n} \to \infty$. When $r_2 = 0$, we put $n = n_1$, i.e., $p := p_\mathbf{n} = 1$. For an integer k, denote by $\mathcal{S}_{m,k}$ the set of all k-subsets of $\{1, \dots, m\}$. For $S \in \mathcal{S}_{n_1,r_1}$, $T \in \mathcal{S}_{n_2,r_2}$, and $i \in S$, define

$$\psi_\mathbf{n}^{(i)}(S;T) = \psi_{\mathbf{n},\mathbf{r},\epsilon}^{(i)}(S;T) := I(d(X_i, X_j) \leq \epsilon, d(X_i, Y_k) \leq \epsilon, \forall j \in S, \forall k \in T),$$

i.e., the indicator of the event that all elements in $\{X_j, j \in S\}$ and $\{Y_k, k \in T\}$ are ϵ-close to X_i. Note that by conditioning we have

$$\mathrm{E}\psi_\mathbf{n}^{(i)}(S;T) = \mathrm{E}p_{X,\epsilon}(X)^{r_1-1} p_{Y,\epsilon}(X)^{r_2} =: q_{\mathbf{r},\epsilon},$$

say, the ϵ-coincidence probability. Let a generalized U-statistic for the functional $q_{\mathbf{r},\epsilon}$ be defined as

$$Q_{\mathbf{n},\mathbf{r}} = Q_{\mathbf{n},\mathbf{r},\epsilon} := \binom{n_1}{r_1}^{-1} \binom{n_2}{r_2}^{-1} \sum_{S \in \mathcal{S}_{n_1,r_1}} \sum_{T \in \mathcal{S}_{n_2,r_2}} \psi_\mathbf{n}(S;T),$$

where the symmetrized kernel

$$\psi_\mathbf{n}(S;T) = \psi_{\mathbf{n},\mathbf{r},\epsilon}(S;T) := \frac{1}{r_1} \sum_{i \in S} \psi_\mathbf{n}^{(i)}(S;T),$$

and by definition, $Q_{\mathbf{n},\mathbf{r}}$ is an unbiased estimator of $q_{\mathbf{r},\epsilon} = \mathrm{E}Q_{\mathbf{n},\mathbf{r}}$. Define for discrete and continuous distributions

$$\begin{aligned}
\zeta_{1,0} = \zeta_{1,0,\mathbf{r}} &:= \mathrm{Var}(p_X(X)^{r_1-1} p_Y(X)^{r_2}) = q_{2r_1-1,2r_2} - q_{r_1,r_2}^2, \\
\zeta_{0,1} = \zeta_{0,1,\mathbf{r}} &:= \mathrm{Var}(p_X(Y)^{r_1} p_Y(Y)^{r_2-1}) = q_{2r_1,2r_2-1} - q_{r_1,r_2}^2, \quad r_2 \geq 1,
\end{aligned}$$

and for the one- and two sample case, let

$$\kappa_\mathbf{r} := \begin{cases} p^{-1} r_1^2 \zeta_{1,0} + (1-p)^{-1} r_2^2 \zeta_{0,1}, & r_2 \geq 1, \\ r_1^2 \zeta_{1,0}, & r_2 = 0. \end{cases}$$

Denote by $\xrightarrow{\mathrm{D}}$ and $\xrightarrow{\mathrm{P}}$ convergence in distribution and in probability, respectively.

The paper is organized as follows. In Section 2, we consider estimation of Rényi entropy functionals for discrete and continuous distributions. In Section 3, we discuss some applications of the obtained estimators in average case analysis for random databases (e.g., for join optimization with approximate matching), in pattern and image matching problems, and for some distribution identification problems. Several numerical experiments demonstrate the rate of convergence in the obtained asymptotic results. Section 4 contains the proofs of the statements from the previous sections.

2 Main Results

2.1 Discrete Distributions

In the discrete case, set $\epsilon = 0$, i.e., exact coincidences are considered. Then $Q_{\mathbf{n},\mathbf{r}}$ is an unbiased estimator of the ϵ-coincidence probability

$$q_{\mathbf{r},0} = q_{\mathbf{r}} = \mathrm{E}I(X_1 = X_i = Y_j, i = 2, \dots, r_1, j = 1, \dots, r_2) = \mathrm{E}p_X(X)^{r_1-1}p_Y(X)^{r_2}.$$

Let $Q_{\mathbf{n},\mathbf{r}} := Q_{\mathbf{n},\mathbf{r},0}$ and

$$K_{\mathbf{n},\mathbf{r}} := \begin{cases} p_{\mathbf{n}}^{-1}r_1^2(Q_{\mathbf{n},2r_1-1,2r_2} - Q_{\mathbf{n},\mathbf{r}}^2) + (1-p_{\mathbf{n}})^{-1}r_2^2(Q_{\mathbf{n},2r_1,2r_2-1} - Q_{\mathbf{n},\mathbf{r}}^2), & r_2 \geq 1, \\ r_1^2(Q_{\mathbf{n},2r_1-1,2r_2} - Q_{\mathbf{n},\mathbf{r}}^2), & r_2 = 0, \end{cases}$$

and $k_{\mathbf{n},\mathbf{r}} := \max(K_{\mathbf{n},\mathbf{r}}, 1/n)$, an estimator of $\kappa_{\mathbf{r}}$. Denote by $H_{\mathbf{n},\mathbf{r}} := \log(\max(Q_{\mathbf{n},\mathbf{r}}, 1/n))/(1-r)$, an estimator of $h_{\mathbf{r}}$.

Remark 1 (Remark). Instead of $1/n$ in the definition of a truncated estimator, a sequence $a_n > 0$, $a_n \to 0$ as as $\mathbf{n} \to \infty$, can be used (cf. Leonenko and Seleznjev, 2010).

The next asymptotic normality theorem for the estimator $Q_{\mathbf{n},\mathbf{r}}$ follows straightforwardly from the general U-statistic theory (see, e.g., Lee, 1990, Koroljuk and Borovskich, 1994) and the Slutsky theorem.

Theorem 1. *If $\kappa_{\mathbf{r}} > 0$, then*

$$\sqrt{n}(Q_{\mathbf{n},\mathbf{r}} - q_{\mathbf{r}}) \xrightarrow{\mathrm{D}} N(0, \kappa_{\mathbf{r}}) \text{ and } \sqrt{n}(Q_{\mathbf{n},\mathbf{r}} - q_{\mathbf{r}})/k_{\mathbf{n},\mathbf{r}}^{1/2} \xrightarrow{\mathrm{D}} N(0,1);$$
$$\sqrt{n}(1-r)Q_{\mathbf{n},\mathbf{r}}(H_{\mathbf{n},\mathbf{r}} - h_{\mathbf{r}})/k_{\mathbf{n},\mathbf{r}}^{1/2} \xrightarrow{\mathrm{D}} N(0,1) \text{ as } \mathbf{n} \to \infty.$$

2.2 Continuous Distributions

In the continuous case, denote by $\tilde{Q}_{\mathbf{n},\mathbf{r}} := Q_{\mathbf{n},\mathbf{r}}/b_\epsilon(d)^{r-1}$ an estimator of $q_{\mathbf{r}}$. Let $\tilde{q}_{\mathbf{r},\epsilon} := \mathrm{E}\tilde{Q}_{\mathbf{n},\mathbf{r}} = q_{\mathbf{r},\epsilon}/b_\epsilon(d)^{r-1}$ and $v_{\mathbf{n}}^2 := \mathrm{Var}(\tilde{Q}_{\mathbf{n},\mathbf{r}})$.

Henceforth, assume that $\epsilon = \epsilon(\mathbf{n}) \to 0$ as $\mathbf{n} \to \infty$. For a sequence of random variables $U_n, n \geq 1$, we say that $U_n = \mathrm{O_P}(1)$ as $n \to \infty$ if for any $\epsilon > 0$ and n large enough there exists $A > 0$ such that $P(|U_n| > A) \leq \epsilon$, i.e., the family of distributions of $U_n, n \geq 1$, is tight, and for a numerical sequence $w_n, n \geq 1$, say, $U_n = \mathrm{O_P}(w_n)$ as $n \to \infty$ if $U_n/w_n = \mathrm{O_P}(1)$ as $n \to \infty$. The following theorem describes the consistency and asymptotic normality properties of the estimator $\tilde{Q}_{\mathbf{n},\mathbf{r}}$.

Theorem 2. *Assume that $p_X(x)$ and $p_Y(x)$ are bounded and continuous or with a finite number of discontinuity points.*

(i) Then $\mathrm{E}\tilde{Q}_{\mathbf{n},\mathbf{r}} \to q_{\mathbf{r}}$ as $\mathbf{n} \to \infty$ and if $n\epsilon^{d(1-1/r)} \to a, 0 < a \leq \infty$, then $v_{\mathbf{n}}^2 = \mathrm{O}(n^{-1}\epsilon^{d(1/r-1)})$. Hence, if $n\epsilon^{d(1-1/r)} \to \infty$, then $\tilde{Q}_{\mathbf{n},\mathbf{r}}$ is a consistent estimator of $q_{\mathbf{r}}$.

(ii) If $\kappa_{\mathbf{r}} > 0$ and $n\epsilon^d \to \infty$, then

$$\sqrt{n}(\tilde{Q}_{\mathbf{n},\mathbf{r}} - \tilde{q}_{\mathbf{r},\epsilon}) \overset{\mathrm{D}}{\to} N(0, \kappa_{\mathbf{r}}) \text{ as } \mathbf{n} \to \infty.$$

In order to evaluate the functional $q_{\mathbf{r}}$, we denote by $H^{(\alpha)}(C), 0 < \alpha \leq 2, C > 0$, a linear space of bounded and continuous in R^d functions satisfying α-Hölder condition if $0 < \alpha \leq 1$ or if $1 < \alpha \leq 2$ with continuous partial derivatives satisfying $(\alpha - 1)$-Hölder condition with constant C. Furthermore, let

$$K_{\mathbf{n},\mathbf{r}} := \begin{cases} p_{\mathbf{n}}^{-1} r_1^2(\tilde{Q}_{\mathbf{n},2r_1-1,2r_2,\epsilon} - \tilde{Q}_{\mathbf{n},\mathbf{r},\epsilon}^2) + (1-p_{\mathbf{n}})^{-1} r_2^2(\tilde{Q}_{\mathbf{n},2r_1,2r_2-1,\epsilon} - \tilde{Q}_{\mathbf{n},\mathbf{r},\epsilon}^2), & r_2 \geq 1, \\ r_1^2(\tilde{Q}_{\mathbf{n},2r_1-1,2r_2,\epsilon} - \tilde{Q}_{\mathbf{n},\mathbf{r},\epsilon}^2), & r_2 = 0, \end{cases}$$

and define $k_{\mathbf{n},\mathbf{r}} := \max(K_{\mathbf{n},\mathbf{r}}, 1/n)$. It follows from Theorem 2 and the Slutsky theorem that $k_{\mathbf{n},\mathbf{r}}$ is a consistent estimator of the asymptotic variance $\kappa_{\mathbf{r}}$. Denote by $H_{\mathbf{n},\mathbf{r}} := \log(\max(\tilde{Q}_{\mathbf{n},\mathbf{r}}, 1/n))/(1-r)$, an estimator of $h_{\mathbf{r}}$. Let $L(n)$ be a slowly varying function. We obtain the following asymptotic result.

Theorem 3. *Let $p_X(x), p_Y(x) \in H^{(\alpha)}(C)$.*
(i) Then the bias $|\tilde{q}_{\mathbf{r},\epsilon} - q_{\mathbf{r}}| \leq C_1\epsilon^\alpha, C_1 > 0$.
(ii) If $0 < \alpha \leq d/2$ and $\epsilon \sim cn^{-\alpha/(2\alpha+d(1-1/r))}, 0 < c < \infty$, then

$$\tilde{Q}_{\mathbf{n},\mathbf{r}} - q_{\mathbf{r}} = \mathrm{O}_{\mathrm{P}}(n^{-\alpha/(2\alpha+d(1-1/r))}) \text{ and } H_{\mathbf{n},\mathbf{r}} - h_{\mathbf{r}} = \mathrm{O}_{\mathrm{P}}(n^{-\alpha/(2\alpha+d(1-1/r))}) \text{ as } \mathbf{n} \to \infty.$$

(iii) If $\kappa_{\mathbf{r}} > 0$, $\alpha > d/2$, $\epsilon \sim L(n)n^{-1/d}$, and $n\epsilon^d \to \infty$, then

$$\sqrt{n}(\tilde{Q}_{\mathbf{n},\mathbf{r}} - q_{\mathbf{r}}) \overset{\mathrm{D}}{\to} N(0, \kappa_{\mathbf{r}}) \text{ and } \sqrt{n}(\tilde{Q}_{\mathbf{n},\mathbf{r}} - q_{\mathbf{r}})/k_{\mathbf{n},\mathbf{r}}^{1/2} \overset{\mathrm{D}}{\to} N(0,1);$$
$$\sqrt{n}(1-r)\tilde{Q}_{\mathbf{n},\mathbf{r}}(H_{\mathbf{n},\mathbf{r}} - h_{\mathbf{r}})/k_{\mathbf{n},\mathbf{r}}^{1/2} \overset{\mathrm{D}}{\to} N(0,1) \text{ as } \mathbf{n} \to \infty.$$

3 Applications and Numerical Experiments

3.1 Approximate Matching in Stochastic Databases

Let tables (in a *relational database*) T_1 and T_2 be matrices with m_1 and m_2 i.i.d. random tuples (or records), respectively. One of basic database operations, *join*, combines two tables into a third one by matching values for given columns (attributes). For example, the join condition can be the equality (equi-join) between a given pairs of attributes (e.g., names) from the tables. Joins are especially important for tieing together pieces of disparate information scattered throughout a database (see, e.g., Kiefer et al. 2005, Copas and Hilton, 1990, and references therein). For the approximate join, we match ϵ-close tuples, say, $d(t_1(j), t_2(i)) \leq \epsilon, t_k(j) \in T_k, k = 1, 2$, with a specified distance, see, e.g., Seleznjev and Thalheim (2008). A set of attributes A in a table T is called an ϵ-key (test) if there are no ϵ-close sub-tuples $t_A(j), j = 1, \ldots, m$. Knowledge about the set of tests (ϵ-keys) is very helpful for avoiding redundancy in identification and searching problems, characterizing the complexity of a database design for further optimization, see, e.g., Thalheim (2000). By joining a table with itself (self-join) we identify also ϵ-keys and key-properties for a set of attributes for a random table (Seleznjev and Thalheim, 2003, Leonenko and Seleznjev, 2010).

The cost of join operations is usually proportional to the size of the intermediate results and so the joining order is a primary target of join-optimizers for multiple (large) tables, Thalheim (2000). The average case approach based on stochastic database modelling for optimization problems is proposed in Seleznjev and Thalheim (2008), where for random databases, the distribution of the ϵ-join size N_ϵ is studied. In particular, with some conditions it is shown that the average size

$$\mathrm{E}N_\epsilon = m_1 m_2 q_{1,1,\epsilon} = m_1 m_2 \epsilon^d b_1(d)(e^{-v} + \mathrm{o}(1)) \text{ as } \epsilon \to 0,$$

that is the asymptotically optimal (in average) pairs of tables are amongst the tables with maximal value of the functional $v = h_{1,1}$ (variability) and the corresponding estimators of $h_{1,1}$ can be used for samples $X_1, \ldots, X_{n_1}$ and $Y_1, \ldots, Y_{n_2}$ from T_1 and T_2, respectively. For discrete distributions, similar results from Theorem 1 for $\epsilon = 0$ can be applied.

3.2 Image Matching Using Entropy Similarity Measures

Image retrieval and registration fall in the general area of pattern matching problems, where the best match to a reference or query image I_0 is to be found in a database of secondary images $\{I_i\}_{i=1}^n$. The best match is expressed as a partial re-indexing of the database in decreasing order of similarity to the reference image using a similarity measure. In the context of image registration, the database corresponds to an infinite set of transformed versions of a secondary image, e.g., rotation and translation, which are compared to the reference image to register the secondary one to the reference.

Let X and Y be d-dimensional random vectors and let $p_X(x)$ and $p_Y(x)$ denote densities for X and Y. In the sequel, X is a feature vector constructed from the query image and Y corresponding feature vector for a secondary image in an image database. When the features are discrete valued the $p_X(x)$ and $p_Y(x)$ are probability mass functions.

The basis for entropy methods of image matching is a measure of similarity between image densities $p_X(x)$ and $p_Y(x)$. A general entropy similarity measure is the Rényi s-divergence, also called the Rényi s-relative entropy,

$$D_s(p_X, p_Y) := \frac{1}{s-1} \log \int_{R^d} p_Y(x) \left(\frac{p_X(x)}{p_Y(x)} \right)^s dx = \frac{1}{s-1} \log \int_{R^d} p_X(x)^s p_Y(x)^{1-s} dx, \quad s \neq 1.$$

When the density $p_X(x)$ is supported on a compact domain and Y is uniformly distributed over this domain, the Rényi s-divergence reduces to the Rényi s-entropy

$$h_s = \frac{1}{1-s} \log \left(\int_{R^d} p_X(x)^s dx \right).$$

Another important example of statistical distance between distributions is given by the following nonsymmetric Bregman distance (see, e.g., Pardo, 2006)

$$B_s = B_s(p_X, p_Y) := \int_{R^d} \left[p_Y(x)^s + \frac{1}{s-1} p_X(x)^s - \frac{s}{s-1} p_X(x) p_Y(x)^{s-1} \right] dx, \quad s \neq 1,$$

or its symmetrized version

$$\begin{aligned} K_s = K_s(p_X, p_Y) &:= \frac{1}{s}[B_s(p_X, p_Y) + B_s(p_Y, p_X)] \\ &= \frac{1}{s-1} \int_{R^d} [p_X(x) - p_Y(x)][p_X(x)^{s-1} - p_Y(x)^{s-1}] dx. \end{aligned}$$

For $s = 2$, we get the second order distance

$$B_2 = K_2 = \int_{R^d} [p_X(x) - p_Y(x)]^2 dx.$$

Observe that, e.g.,

$$B_s = q_{0,s} + q_{s,0}/(s-1) - s q_{1,s-1}/(s-1).$$

So, for an integer s, applying Theorems 1 and 3 one can obtain an asymptotically normal estimator of the Rényi s-entropy and a consistent estimator of the Bregman distance.

3.3 Entropy Maximizing Distributions

For a positive definite and symmetric matrix Σ, $s \neq 1$, define the constants

$$m = d + 2/(s-1), \qquad \mathbf{C}_s = (m+2)\Sigma,$$

and

$$A_s = \frac{1}{|\pi \mathbf{C}_s|^{1/2}} \frac{\Gamma(m/2+1)}{\Gamma((m-d)/2+1)}.$$

Among all densities with mean μ and covariance matrix Σ, the Rényi entropy h_s, $s = 2, \ldots$, is uniquely maximized by the density (Costa et al. 2003)

$$p_s^*(x) = \begin{cases} A_s(1 - (x-\mu)^T \mathbf{C}_s^{-1}(x-\mu))^{1/(s-1)}, & x \in \Omega_s \\ 0, & x \notin \Omega_s, \end{cases} \tag{1}$$

with support

$$\Omega_s = \{x \in R^d : (x-\mu)^T \mathbf{C}_s^{-1}(x-\mu) \leq 1\}.$$

The distribution defined by $p_s^*(x)$ belongs to the class of *Student-r* distributions. Let $\mathcal{K}$ be a class of d-dimensional density functions $p(x)$, $x \in R^d$, with positive definite covariance matrix. By the procedure described in Leonenko and Seleznjev (2010), the proposed estimator of h_s can be used for distribution identification problems, i.e., to test the null hypothesis $H_0 : X_1, \ldots, X_n$ *is a sample from a Student-r distribution of type* (1) against the alternative $H_1 : X_1, \ldots, X_n$ *is a sample from any other member of* $\mathcal{K}$.

3.4 Numerical Experiments

Example 1. Figure 1 shows the accuracy of the estimator for the cubic Rényi entropy h_3 of discrete distributions in Theorem 1, for a sample from a d-dimensional Bernoulli distribution and n observations, $d = 3$, $n = 300$, with Bernoulli $Be(p)$-i.i.d. components, $p = 0.8$. Here the coincidence probability $q_3 = (p^3 + (1-p)^3)^3$ and the Rényi entropy $h_3 = -\log(q_3)/2$. The histogram for the normalized residuals $r_n^{(i)} := 2\sqrt{n}Q_{\mathbf{n},\mathbf{r}}(H_{\mathbf{n},\mathbf{r}} - h_{\mathbf{r}})/k_{\mathbf{n},\mathbf{r}}^{1/2}$, $i = 1, \ldots, N_{sim}$ are compared to the standard normal density, $N_{sim} = 500$. The corresponding qq-plot and p-values for the Kolmogorov-Smirnov (0.4948) and Shapiro-Wilk (0.7292) tests also support normality hypothesis for the obtained residuals.

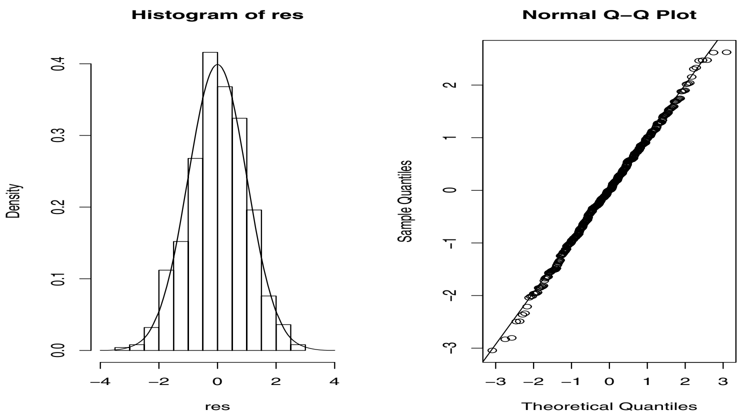


Fig. 1. Bernoulli d-dimensional distribution; $d = 3$, $Be(p)$-i.i.d. components, $p = 0.8$, sample size $n = 200$. Standard normal approximation for the empirical distribution (histogram) for the normalized residuals, $N_{sim} = 500$.

Example 2. Figure 2 illustrates the performance of the approximation for the differentiable variability $v = h_{1,1}$ in Theorem 3, for two one-dimensional samples from normal distributions $N(0, 3/2)$ and $N(2, 1/2)$, with the sample sizes $n_1 = 100, n_2 = 200$, respectively. Here the variability $v = \log(2\sqrt{\pi}e)$. The normalized residuals are compared to the standard normal density, $N_{sim} = 300$. The qq-plot and p-values for the Kolmogorov-Smirnov (0.9916) and Shapiro-Wilk (0.5183) tests also support the normal approximation.

Example 3. Figure 3 shows the accuracy of the normal approximation for the cubic Rényi entropy h_3 in Theorem 3, for a sample from a bivariate Gaussian distribution with $N(0, 1)$-i.i.d. components, and $n = 300$ observations. Here the Rényi entropy $h_3 = \log(\sqrt{12}\pi)$. The histogram, qq-plot, and p-values for the Kolmogorov-Smirnov (0.2107) and Shapiro-Wilk (0.2868) tests allow to accept the hypothesis of standard normality for the residuals, $N_{sim} = 300$.

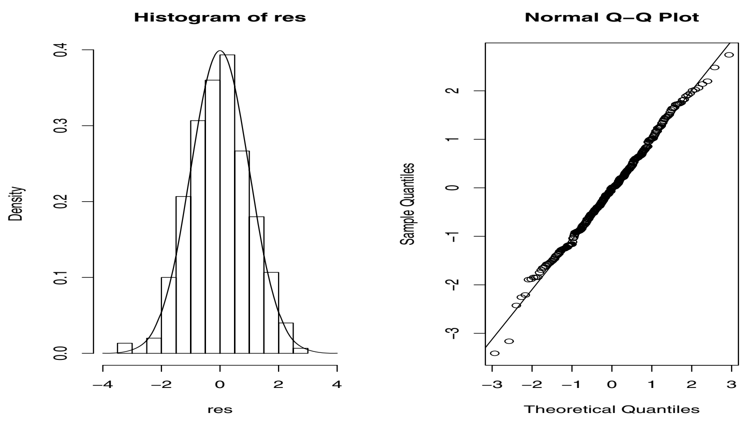


Fig. 2. Two Gaussian distributions; $N(0, 3/2)$, $N(2, 1/2)$, $n_1 = 100, n_2 = 200, \epsilon = 1/10$. Standard normal approximation for the empirical distribution (histogram) for the normalized residuals, $N_{sim} = 300$.

Example 4. Figure 4 demonstrates the behaviour of the estimator for the quadratic Bregman distance $B_2(p_X, p_Y) = 1/2$ for two exponential distributions $p_X(x) = \beta_1 e^{-\beta_1 x}, x > 0$, and $p_Y(x) = \beta_2 e^{-\beta_2 x}, x > 0$, with rate parameters $\beta_1 = 1, \beta_2 = 3$, respectively, and equal sample sizes. The empirical mean squared error (MSE) based on 10000 independent simulations are calculated for different values of n.

4 Proofs

Lemma 1. *Assume that $p_X(x)$ and $p_Y(x)$ are bounded and continuous or with a finite number of discontinuity points. Let $a, b \geq 0$. Then*

$$b_\epsilon(d)^{-(a+b)}\mathrm{E}(p_{X,\epsilon}(X)^a p_{Y,\epsilon}(X)^b) \to \int_{R^d} p_X(x)^{a+1} p_Y(x)^b dx \text{ as } \epsilon \to 0.$$

Proof: We have

$$b_\epsilon(d)^{-(a+b)}\mathrm{E}(p_{X,\epsilon}(X)^a p_{Y,\epsilon}(X)^b) = \mathrm{E}(g_\epsilon(X)),$$

where $g_\epsilon(x) := (p_{X,\epsilon}(x)/b_\epsilon(d))^a (p_{Y,\epsilon}(x)/b_\epsilon(d))^b$. It follows by definition that $g_\epsilon(x) \to p_X(x)^a p_Y(x)^b$ as $\epsilon \to 0$, for all continuity points of $p_X(x)$ and $p_Y(x)$, and that the random variable $g_\epsilon(X)$ is bounded. Hence, the bounded convergence theorem implies

$$\mathrm{E}(g_\epsilon(X)) \to \mathrm{E}(p_X(X)^a p_Y(X)^b) = \int_{R^d} p_X(x)^{a+1} p_Y(x)^b dx \text{ as } \epsilon \to 0.$$

$\square$

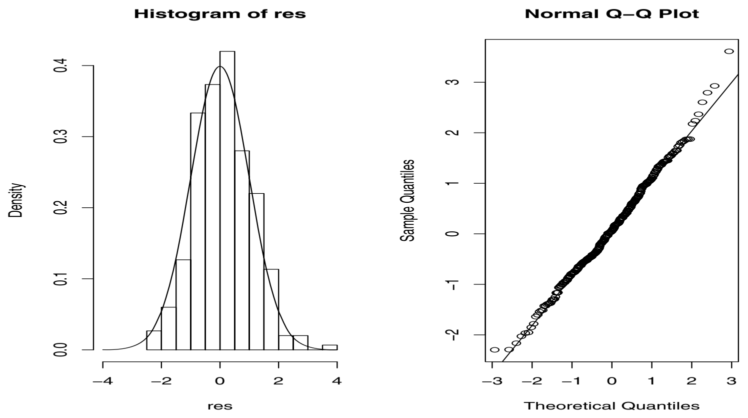


Fig. 3. Bivariate normal distribution with $N(0,1)$-i.i.d. components; sample size $n = 300, \epsilon = 1/2$. Standard normal approximation for the empirical distribution (histogram) for the normalized residuals, $N_{sim} = 300$.

Proof of Theorem 2: (i) The asymptotic unbiasedness $\mathrm{E}\tilde{Q}_{\mathbf{n},\mathbf{r}} \to q_{\mathbf{r}}$ follows from Lemma 1. Note that we can assume without loss of generality that $\zeta_{1,0} > 0$. Recall that we always have $r_1 \geq 1$. In addition, first let $r_2 \geq 1$ and $\zeta_{0,1} > 0$. We use the conventional results from the theory of U-statistics (see, e.g., Lee, 1990, Koroljuk and Borovskich, 1994). For $l = 0, \dots, r_1$, and $m = 0, \dots, r_2$, define

$$\begin{aligned}\psi_{l,m,\mathbf{n}}(x_1,\dots,x_l;y_1,\dots,y_m) &:= \mathrm{E}\psi_{\mathbf{n}}(x_1,\dots,x_l,X_{l+1},\dots,X_{r_1};y_1,\dots,y_m,Y_{m+1},\dots,Y_{r_2})\\ &= \frac{1}{r_1}\sum_{i=1}^{r_1}\mathrm{E}\psi_{\mathbf{n}}^{(i)}(x_1,\dots,x_l,X_{l+1},\dots,X_{r_1};y_1,\dots,y_m,Y_{m+1},\dots,Y_{r_2}), \qquad (2)\end{aligned}$$

and

$$\sigma^2_{l,m,\epsilon} := \mathrm{Var}(\psi_{l,m,\mathbf{n}}(X_1,\dots,X_l;Y_1,\dots,Y_m)).$$

Let $S_1, S_2 \in \mathcal{S}_{n_1,r_1}$ and $T_1, T_2 \in \mathcal{S}_{n_2,r_2}$ have l and m elements in common, respectively. By properties of U-statistics, we have

$$v^2_{\mathbf{n}} = \mathrm{Var}(\tilde{Q}_{\mathbf{n},\mathbf{r}}) = b_\epsilon(d)^{-2(r-1)}\sum_{l=0}^{r_1}\sum_{m=0}^{r_2}\frac{\binom{r_1}{l}\binom{r_2}{m}\binom{n_1-r_1}{r_1-l}\binom{n_2-r_2}{r_2-m}}{\binom{n_1}{r_1}\binom{n_2}{r_2}}\sigma^2_{l,m,\epsilon}, \qquad (3)$$

and

$$\sigma^2_{l,m,\epsilon} = \mathrm{Cov}(\psi_{\mathbf{n}}(S_1;T_1), \psi_{\mathbf{n}}(S_2;T_2)). \qquad (4)$$

From (4) we get that $0 \leq \sigma^2_{l,m,\epsilon} \leq \mathrm{E}(\psi_{\mathbf{n}}(S_1;T_1)\psi_{\mathbf{n}}(S_2;T_2))$, which is a finite linear combination of $P(A_i^{(1)} \cap A_j^{(2)}), i \in S_1, j \in S_2$, where, for $u = 1, 2$,

$$A_i^{(u)} := \{d(X_i,X_k) \leq \epsilon, d(X_i,Y_s) \leq \epsilon, \forall k \in S_u, \forall s \in T_u\}, \quad i \in S_u.$$

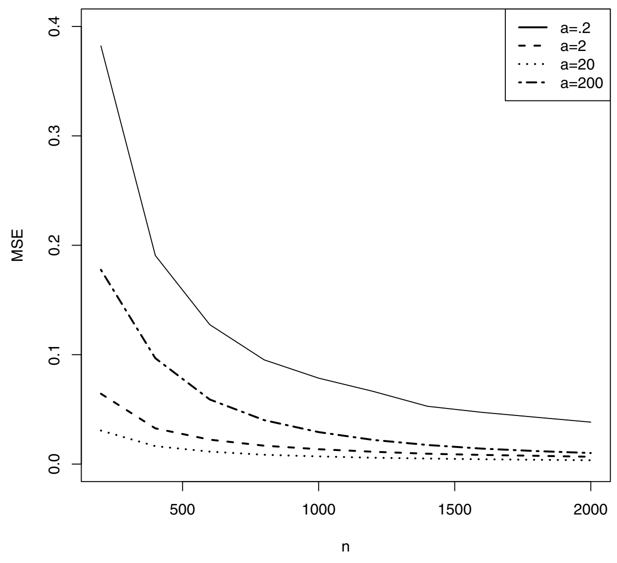


Fig. 4. Bregman distance for $Exp(\beta_1)$ and $Exp(\beta_2)$, $\beta_1 = 1, \beta_2 = 3$. The empirical MSE obtained for the U-statistic estimator with $n\epsilon = a$, for different values of a.

When $l \neq 0$ or $m \neq 0$, the triangle inequality implies that

$$A_i^{(1)} \cap A_j^{(2)} \subseteq F_i := \{d(X_i, X_k) \leq 3\epsilon, d(X_i, Y_s) \leq 3\epsilon, \forall k \in S_1 \cup S_2, \forall s \in T_1 \cup T_2\}.$$

Since $|S_1 \cup S_2| = 2r_1 - l$ and $|T_1 \cup T_2| = 2r_2 - m$, it follows by conditioning and from Lemma 1 that

$$\begin{aligned} P(A_i^{(1)} \cap A_j^{(2)}) \leq P(F_i) &= \mathrm{E}(p_{X,3\epsilon}(X_i)^{2r_1-l-1} p_{Y,3\epsilon}(X_i)^{2r_2-m}) \\ &\sim 3^{d(2r-l-m-1)} b_\epsilon(d)^{2r-l-m-1} q_{2r_1-l,2r_2-m} \text{ as } \mathbf{n} \to \infty. \end{aligned}$$

We conclude that

$$\sigma^2_{l,m,\epsilon} = \mathrm{O}(b_\epsilon(d)^{2r-l-m-1}) \text{ as } \mathbf{n} \to \infty. \tag{5}$$

Now, for $l = 0, \ldots, r_1$, $m = 0, \ldots, r_2$, and some constant $C_{l.m}$, (5) implies that

$$\begin{aligned} b_\epsilon(d)^{-2(r-1)} \frac{\binom{r_1}{l}\binom{r_2}{m}\binom{n_1-r_1}{r_1-l}\binom{n_2-r_2}{r_2-m}}{\binom{n_1}{r_1}\binom{n_2}{r_2}} \sigma^2_{l,m,\epsilon} &\sim C_{l,m} \frac{b_\epsilon(d)^{-(2r-l-m-1)} \sigma^2_{l,m,\epsilon}}{n^{l+m} \epsilon^{d(l+m-1)}} \\ &= \mathrm{O}\left(\frac{1}{n^{l+m}\epsilon^{d(l+m-1)}}\right) \text{ as } \mathbf{n} \to \infty, \end{aligned} \tag{6}$$

and note that, for $k = 1, \ldots, r$,

$$n^k \epsilon^{d(k-1)} = (n\epsilon^{d(1-1/k)})^k \geq (n\epsilon^{d(1-1/r)})^k. \tag{7}$$

Hence, since $n\epsilon^{d(1-1/r)} \to a, 0 < a \le \infty,$ as $\mathbf{n} \to \infty$, it follows from (3), (6), and (7), that $v_{\mathbf{n}}^2 = \mathrm{O}((n\epsilon^{d(1-1/r)})^{-1})$. So, when $n\epsilon^{d(1-1/r)} \to \infty$, we get that

$$\mathrm{E}(\tilde{Q}_{\mathbf{n},\mathbf{r}} - q_{\mathbf{r}})^2 = v_{\mathbf{n}}^2 + (\mathrm{E}(\tilde{Q}_{\mathbf{n},\mathbf{r}} - q_{\mathbf{r}}))^2 \to 0 \text{ as } \mathbf{n} \to \infty,$$

and the assertion follows.
(ii) Let

$$h_{\mathbf{n},\mathbf{r}}^{(1,0)}(x) := \psi_{1,0,\mathbf{n}}(x)/b_\epsilon(d)^{r-1} - \tilde{q}_{\mathbf{r},\epsilon}, \quad h_{\mathbf{n},\mathbf{r}}^{(0,1)}(x) := \psi_{0,1,\mathbf{n}}(x)/b_\epsilon(d)^{r-1} - \tilde{q}_{\mathbf{r},\epsilon}.$$

The H-decomposition of $\tilde{Q}_{\mathbf{n},\mathbf{r}}$ is given by

$$\tilde{Q}_{\mathbf{n},\mathbf{r}} = \tilde{q}_{\mathbf{r},\epsilon} + r_1 H_{\mathbf{n},\mathbf{r}}^{(1,0)} + r_2 H_{\mathbf{n},\mathbf{r}}^{(0,1)} + R_{\mathbf{n}}, \tag{8}$$

where

$$H_{\mathbf{n},\mathbf{r}}^{(1,0)} := \frac{1}{n_1}\sum_{i=1}^{n_1} h_{\mathbf{n},\mathbf{r}}^{(1,0)}(X_i), \quad H_{\mathbf{n},\mathbf{r}}^{(0,1)} := \frac{1}{n_2}\sum_{i=1}^{n_2} h_{\mathbf{n},\mathbf{r}}^{(0,1)}(Y_i).$$

The terms in (8) are uncorrelated, and since $\mathrm{Var}(h_{\mathbf{n},\mathbf{r}}^{(1,0)}(X_1)) = b_\epsilon(d)^{-2(r-1)}\sigma_{1,0,\epsilon}^2$ and $\mathrm{Var}(h_{\mathbf{n},\mathbf{r}}^{(0,1)}(Y_1)) = b_\epsilon(d)^{-2(r-1)}\sigma_{0,1,\epsilon}^2$, we obtain from (3) that

$$\begin{aligned}
\mathrm{Var}(R_{\mathbf{n}}) &= \mathrm{Var}(\tilde{Q}_{\mathbf{n},\mathbf{r}}) - \mathrm{Var}(r_1 H_{\mathbf{n},\mathbf{r}}^{(1,0)}) - \mathrm{Var}(r_2 H_{\mathbf{n},\mathbf{r}}^{(0,1)}) \\
&= \mathrm{Var}(\tilde{Q}_{\mathbf{n},\mathbf{r}}) - b_\epsilon(d)^{-2(r-1)} r_1^2 n_1^{-1}\sigma_{1,0,\epsilon}^2 - b_\epsilon(d)^{-2(r-1)} r_2^2 n_2^{-1}\sigma_{0,1,\epsilon}^2 \\
&= K_{1,\mathbf{n}} b_\epsilon(d)^{-2(r-1)} n^{-1}\sigma_{1,0,\epsilon}^2 + K_{2,\mathbf{n}} b_\epsilon(d)^{-2(r-1)} n^{-1}\sigma_{0,1,\epsilon}^2 \\
&\quad + b_\epsilon(d)^{-2(r-1)} \sum_E \frac{\binom{r_1}{l}\binom{r_2}{m}\binom{n_1-r_1}{r_1-l}\binom{n_2-r_2}{r_2-m}}{\binom{n_1}{r_1}\binom{n_2}{r_2}}\sigma_{l,m,\epsilon}^2,
\end{aligned} \tag{9}$$

where $E := \{(l,m) : 0 \le l \le r_1, 0 \le m \le r_2, l+m \ge 2\}$, and

$$K_{1,\mathbf{n}} := r_1^2 p_{\mathbf{n}} \left(\frac{\binom{n_1-r_1}{r_1-1}\binom{n_2-r_2}{r_2}}{\binom{n_1-1}{r_1-1}\binom{n_2}{r_2}} - 1 \right), \quad K_{2,\mathbf{n}} := r_2^2(1-p_{\mathbf{n}}) \left(\frac{\binom{n_1-r_1}{r_1}\binom{n_2-r_2}{r_2-1}}{\binom{n_1}{r_1}\binom{n_2-1}{r_2-1}} - 1 \right).$$

Note that $K_{1,\mathbf{n}}, K_{2,\mathbf{n}} = \mathrm{O}(n^{-1})$ as $\mathbf{n} \to \infty$ so if $n\epsilon^d \to a$, $0 < a \le \infty$, then (5), (6), and (9) imply that $\mathrm{Var}(R_{\mathbf{n}}) = \mathrm{O}((n^2\epsilon^d)^{-1})$ as $\mathbf{n} \to \infty$. In particular, for $a = \infty$,

$$\mathrm{Var}(R_{\mathbf{n}}) = \mathrm{o}(n^{-1}) \Rightarrow n^{1/2} R_{\mathbf{n}} \xrightarrow{\mathrm{P}} 0 \text{ as } \mathbf{n} \to \infty. \tag{10}$$

By symmetry, we have from (2)

$$\psi_{1,0,\mathbf{n}}(x) = \frac{1}{r_1}\Big(p_{X,\epsilon}(x)^{r_1-1} p_{Y,\epsilon}(x)^{r_2} + (r_1-1)\mathrm{E}(\psi_{\mathbf{n}}^{(2)}(x, X_2, \ldots, X_{r_1}; Y_1, \ldots, Y_{r_2}))\Big). \tag{11}$$

Let x be a continuity point of $p_X(x)$ and $p_Y(x)$. Then, changing variables $y = x + \epsilon u$ and the bounded convergence theorem give

$$\begin{aligned}
\mathrm{E}(\psi_{\mathbf{n}}^{(2)}(x, X_2, \ldots, X_{r_1}; Y_1, \ldots, Y_{r_2}) &= \mathrm{E}(\mathrm{E}(\psi_{\mathbf{n}}^{(2)}(x, X_2, \ldots, X_{r_1}; Y_1, \ldots, Y_{r_2})|X_2)) \\
&= \int_{R^d} I(d(x,y) \leq \epsilon) p_{X,\epsilon}(y)^{r_1-2} p_{Y,\epsilon}(y)^{r_2} p_X(y) dy \\
&= \epsilon^d \int_{R^d} I(d(0,u) \leq 1) p_{X,\epsilon}(x+\epsilon u)^{r_1-2} p_{Y,\epsilon}(x+\epsilon u)^{r_2} p_X(x+\epsilon u) du \qquad (12) \\
&\sim b_\epsilon(d)^{r-1} p_X(x)^{r_1-1} p_Y(x)^{r_2} \text{ as } \mathbf{n} \to \infty.
\end{aligned}$$

From (11) we get that

$$\psi_{1,0,\mathbf{n}}(x) \sim b_\epsilon(d)^{r-1} p_X(x)^{r_1-1} p_Y(x)^{r_1} \text{ as } \mathbf{n} \to \infty,$$

and hence

$$\lim_{\mathbf{n}\to\infty} h_{\mathbf{n},\mathbf{r}}^{(1,0)}(x) = p_X(x)^{r_1-1} p_Y(x)^{r_2} - q_{\mathbf{r}}, \qquad (13)$$

and similarly,

$$\lim_{\mathbf{n}\to\infty} h_{\mathbf{n},\mathbf{r}}^{(0,1)}(x) = p_X(x)^{r_1} p_Y(x)^{r_2-1} - q_{\mathbf{r}}. \qquad (14)$$

Let $\max(p_X(x), p_Y(x)) \leq C, x \in R^d$. Then $\max(p_{X,\epsilon}(x), p_{Y,\epsilon}(x)) \leq b_\epsilon(d)C, x \in R^d$. It follows from (11) and (12) that $\psi_{1,0,\mathbf{n}}(x) \leq b_\epsilon(d)^{r-1}C^{r-1}, x \in R^d$, and hence $h_{\mathbf{n},\mathbf{r}}^{(1,0)}(x) \leq 2C^{r-1}$, $x \in R^d$. Similarly, we have that $h_{\mathbf{n},\mathbf{r}}^{(0,1)}(x) \leq 2C^{r-1}$, $x \in R^d$. Therefore, $h_{\mathbf{n},\mathbf{r}}^{(1,0)}(X_1)$ and $h_{\mathbf{n},\mathbf{r}}^{(0,1)}(Y_1)$ are bounded random variables. Hence, from (13), (14), and the bounded convergence theorem we obtain

$$\mathrm{Var}(h_{\mathbf{n},\mathbf{r}}^{(1,0)}(X_1)) \to \zeta_{1,0}, \quad \mathrm{Var}(h_{\mathbf{n},\mathbf{r}}^{(0,1)}(Y_1)) \to \zeta_{0,1} \text{ as } \mathbf{n} \to \infty.$$

Let $Z_{\mathbf{n},i} := n_1^{-1/2} h_{\mathbf{n},\mathbf{r}}^{(1,0)}(X_i)$, $i = 1, \ldots, n_1$, and observe that, for $\delta > 0$,

$$\sum_{i=1}^{n_1} \mathrm{E}Z_{\mathbf{n},i}^2 = \mathrm{Var}(h_{\mathbf{n},\mathbf{r}}^{(1,0)}(X_1)) \to \zeta_{1,0} > 0 \text{ as } \mathbf{n} \to \infty,$$

$$\begin{aligned}
\lim_{\mathbf{n}\to\infty} \sum_{i=1}^{n_1} \mathrm{E}(|Z_{\mathbf{n},i}|^2 I(|Z_{\mathbf{n},i}| > \delta)) &= \lim_{\mathbf{n}\to\infty} \mathrm{E}(|h_{\mathbf{n},\mathbf{r}}^{(1,0)}(X_1)|^2 I(|h_{\mathbf{n},\mathbf{r}}^{(1,0)}(X_1)| > \delta n_1^{1/2})) \\
&\leq \lim_{\mathbf{n}\to\infty} 4C^{2(r-1)} \mathrm{E}(I(|h_{\mathbf{n},\mathbf{r}}^{(1,0)}(X_1)| > \delta n_1^{1/2})) = 0,
\end{aligned}$$

where the last equality follows from the boundedness of $h_{\mathbf{n},\mathbf{r}}^{(1,0)}(X_1)$. The Lindeberg-Feller theorem (see, e.g., Theorem 4.6, Durrett, 1991) gives that

$$Z_{\mathbf{n},1} + \ldots + Z_{\mathbf{n},n_1} = n_1^{1/2} H_{\mathbf{n},\mathbf{r}}^{(1,0)} \xrightarrow{\mathrm{D}} N(0, \zeta_{1,0}) \text{ as } \mathbf{n} \to \infty,$$

and similarly $n_2^{1/2}H^{(0,1)}_{\mathbf{n},\mathbf{r}} \overset{\mathrm{D}}{\to} N(0,\zeta_{0,1})$ as $\mathbf{n}\to\infty$. Hence, by independence we get that

$$\begin{aligned}&n^{1/2}(r_1H^{(1,0)}_{\mathbf{n},\mathbf{r}}+r_2H^{(0,1)}_{\mathbf{n},\mathbf{r}})\\&=\frac{r_1}{p_{\mathbf{n}}^{1/2}}n_1^{1/2}H^{(1,0)}_{\mathbf{n},\mathbf{r}}+\frac{r_2}{(1-p_{\mathbf{n}})^{1/2}}n_2^{1/2}H^{(0,1)}_{\mathbf{n},\mathbf{r}} \overset{\mathrm{D}}{\to} N(0,\kappa_{\mathbf{r}}) \text{ as } \mathbf{n}\to\infty.\end{aligned}$$

So, (10) and the Slutsky theorem imply

$$\begin{aligned}&n^{1/2}(\tilde{Q}_{\mathbf{n},\mathbf{r}}-\tilde{q}_{\mathbf{r},\epsilon})\\&=n^{1/2}(r_1H^{(1,0)}_{\mathbf{n},\mathbf{r}}+r_2H^{(0,1)}_{\mathbf{n},\mathbf{r}})+n^{1/2}R_{\mathbf{n}} \overset{\mathrm{D}}{\to} N(0,\kappa_{\mathbf{r}}) \text{ as } \mathbf{n}\to\infty.\end{aligned}$$

Corresponding results for the one-sample case $r_2=0$ and when $\zeta_{0,1}=0$ follow in a similar way. This completes the proof. □

Proof of Theorem 3: The proof is similar to that of the corresponding result in Leonenko and Seleznjev (2010) so we give the main steps only. First we evaluate the bias term $B_{\mathbf{n}} := \tilde{q}_{\mathbf{r},\epsilon}-q_{\mathbf{r}}$. Let $V := (V_1,\ldots,V_d)'$ be an auxiliary random vector uniformly distributed in the unit ball $B_1(0)$, say, $V\in U(B_1(0))$. Then by definition, we have

$$B_{\mathbf{n}}=\int_{R^d}p_{X,\epsilon}(x)^{r_1-1}p_{Y,\epsilon}(x)^{r_2}p_X(x)dx-\int_{R^d}p_X(x)^{r_1}p_Y(x)^{r_2}dx=\mathrm{E}(D_\epsilon(X)),$$

where

$$\begin{aligned}D_\epsilon(x) &:= p_{X,\epsilon}(x)^{r_1-1}p_{Y,\epsilon}(x)^{r_2}-p_X(x)^{r_1-1}p_Y(x)^{r_2}\\&= p_{X,\epsilon}(x)^{r_1-1}(p_{Y,\epsilon}(x)^{r_2}-p_Y(x)^{r_2})+p_Y(x)^{r_2}(p_{X,\epsilon}(x)^{r_1-1}-p_X(x)^{r_1-1}).\end{aligned}$$

It follows by definition that

$$\begin{aligned}D_\epsilon(x) &= P_1(x)(p_{Y,\epsilon}(x)-p_Y(x))+P_2(x)(p_{X,\epsilon}(x)-p_X(x))\\&= \mathrm{E}\big(P_1(x)(p_Y(x-\epsilon V)-p_Y(x))+P_2(x)(p_X(x-\epsilon V)-p_X(x))\big)\end{aligned}$$

where $P_1(x)$ and $P_2(x)$ are polynomials in $p_X(x), p_Y(x), \mathrm{E}(p_X(x-\epsilon V))$, and $\mathrm{E}(p_Y(x-\epsilon V))$. Now the boundedness of $p_X(x)$ and $p_Y(x)$ and the Hölder condition for the continuous differentiable cases imply

$$|D_\epsilon(x)|\le CC_1\epsilon^\alpha, C_1>0,$$

and the assertion (i) follows.

For $\epsilon\sim cn^{-1/(2\alpha+d(1-1/r))}, 0<c<\infty$, $\alpha<d/2$, by (i) and Theorem 1, we have

$$B_{\mathbf{n}}^2+v_{\mathbf{n}}^2=\mathrm{O}(n^{-2\alpha/(2\alpha+d(1-1/r))}).$$

Now for some $C>0$ and any $A>0$ and large enough n_1,n_2, we obtain

$$P\left(|\tilde{Q}_{\mathbf{n},\mathbf{r}}-q_{\mathbf{r}}|>An^{-\alpha/(2\alpha+d(1-1/r))}\right)\le n^{2\alpha/(2\alpha+d(1-1/r))}\frac{B_{\mathbf{n}}^2+v_{\mathbf{n}}^2}{A^2}\le\frac{C}{A^2},$$

and the assertion (ii) follows. Similarly for $\alpha=d/2$.

Finally, for $\alpha>d/2$, $\epsilon\sim L(n)n^{-1/d}$, and $n\epsilon^d\to\infty$, the assertion (iii) follows from Theorem 2 and the Slutsky theorem. This completes the proof. □

Acknowledgment. The third author is partly supported by the Swedish Research Council grant 2009-4489 and the project "Digital Zoo" funded by the European Regional Development Fund. The second author is partly supported of the Commissions the European Communities grant PIRSES-GA 2008-230804 "Marie Curie Actions".

References

Baryshnikov, Y., Penrose, M., Yukich, J.E.: Gaussian limits for generalized spacings. Ann. Appl. Probab. 19, 158–185 (2009)

Copas, J.B., Hilton, F.J.: Record linkage: statistical models for matching computer records. Jour. Royal Stat. Soc. Ser. A 153, 287–320 (1990)

Costa, J., Hero, A., Vignat, C.: On Solutions to Multivariate Maximum α-entropy Problems. In: Rangarajan, A., Figueiredo, M.A.T., Zerubia, J. (eds.) EMMCVPR 2003. LNCS, vol. 2683, pp. 211–228. Springer, Heidelberg (2003)

Demetrovics, J., Katona, G.O.H., Miklós, D., Seleznjev, O., Thalheim, B.: The Average Length of Keys and Functional Dependencies in (Random) Databases. In: Vardi, M.Y., Gottlob, G. (eds.) ICDT 1995. LNCS, vol. 893, pp. 266–279. Springer, Heidelberg (1995)

Demetrovics, J., Katona, G.O.H., Miklós, D., Seleznjev, O., Thalheim, B.: Asymptotic properties of keys and functional dependencies in random databases. Theor. Computer Science 190, 151–166 (1998)

Demetrovics, J., Katona, G.O.H., Miklós, D., Seleznjev, O., Thalheim, B.: Functional dependencies in random databases. Studia Scien. Math. Hungarica 34, 127–140 (1998)

Durrett, R.: Probability: Theory and Examples. Brooks/Cole Publishing Company, New York (1991)

Kapur, J.N.: Maximum-entropy Models in Science and Engineering. Wiley, New York (1989)

Kapur, J.N., Kesavan, H.K.: Entropy Optimization Principles with Applications. Academic Press, New York (1992)

Kiefer, M., Bernstein, A., Lewis, P.M.: Database Systems: An Application-Oriented Approach. Addison-Wesley (2005)

Koroljuk, V.S., Borovskich, Y.V.: Theory of U-statistics. Kluwer, London (1994)

Kozachenko, L.F., Leonenko, N.N.: On statistical estimation of entropy of random vector. Problems Infor. Transmiss. 23, 95–101 (1987)

Lee, A.J.: U-Statistics: Theory and Practice. Marcel Dekker, New York (1990)

Leonenko, N., Pronzato, L., Savani, V.: A class of Rényi information estimators for multidimensional densities. Ann. Stat. 36, 2153–2182 (2008); Corrections, Ann. Stat. 38(6), 3837–3838 (2010)

Leonenko, N., Seleznjev, O.: Statistical inference for the ϵ-entropy and the quadratic Rényi entropy. Jour. Multivariate Analysis 101, 1981–1994 (2010)

Pardo, L.: Statistical Inference Based on Divergence Measures. Chapman and Hall (2006)

Rényi, A.: On measures of entropy and information. In: Proc. 4th Berkeley Symp. Math. Statist. Prob., vol. 1 (1961)

Rényi, A.: Probability Theory. North-Holland, London (1970)

Seleznjev, O., Thalheim, B.: Average case analysis in database problems. Methodol. Comput. Appl. Prob. 5, 395–418 (2003)

Seleznjev, O., Thalheim, B.: Random databases with approximate record matching. Methodol. Comput. Appl. Prob. 12, 63–89 (2008), doi:10.1007/s11009-008-9092-4 (2010) (published online, in print)

Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. Jour. 27, 379–423, 623–656 (1948)

Szpankowski, W.: Average Case Analysis of Algorithms on Sequences. John Wiley, New York (2001)

Thalheim, B.: Entity-Relationship Modeling. In: Foundations of Database Technology. Springer, Berlin (2000)

Tsybakov, A.B., Van der Meulen, E.C.: Root-n consistent estimators of entropy for densities with unbounded support. Scandinavian Jour. Statistics 23, 75–83 (1996)

The Subject-Oriented Approach to Software Design and the Abstract State Machines Method

Egon Börger

Università di Pisa, Dipartimento di Informatica, I-56125 Pisa, Italy
boerger@di.unipi.it

Abstract. In [32, Appendix] we have shown that the system which implements the Subject-oriented approach to Business Process Modeling (S-BPM) has a precise semantical foundation in terms of Abstract State Machines (ASMs). The construction of an ASM model for the basic S-BPM concepts revealed a strong relation between S-BPM and the ASM method for software design and analysis. In this paper we investigate this relation more closely. We use the analysis to evaluate S-BPM as an approach to business process modeling and to suggest some challenging practical extension of the S-BPM system.

1 Introduction

The recent book [32] on the Subject-oriented approach to Business Process Modeling (S-BPM) contains a precise high-level definition, namely in terms of Abstract State Machines (ASMs), of the semantics of business process models developed using the S-BPM tool environment.[1] The construction of an ASM which rigorously describes the basic S-BPM concepts revealed an intimate relation between on the one side S-BPM, whose conceptual origins go back to Fleischmann's software engineering book [30, Part II], and on the other side the ASM method [26], a systems engineering method which too has been developed in the 90'ies of the last century by a community effort building upon Gurevich's discovery of the notion of ASM [40] (at the time called by various names, in 1994 'evolving algebras', for the historical details see [11] or [26, Ch.9]).

In this paper we investigate the striking methodological and conceptual similarities (Sect. 2) and some differences (Sect. 3) of these two independent developments. We propose to enhance the current S-BPM system by offering the modeler tool support for the use of the full ASM-refinement method which generalizes the refinement scheme S-BPM provides the software engineer with.

[1] In the appendix, which is written in English, an ASM interpreter is defined for the behavior of such business process models. The software used to transform the pdf-file generated from latex sources into a Word document and printer-control-compatible format produced a certain number of partly annoying, partly misleading mistakes in the printed text. The interested reader can download the pdf-file for the correct text from [63].

We use this analysis to evaluate S-BPM in terms of six well-known principles for reliable software development (Sect. 4), an evaluation which shows that S-BPM provides practitioners with suitable means to precisely and faithfully *capture* business scenarios and *analyze*, *communicate* and *manage* the resulting models.[2]

What nowadays is called S-BPM is really a version tailored for the development of business processes (BPs) of a more general subject-oriented software engineering method and environment for the development of concurrent systems proposed in [30, Part II] and called there SAPP/PASS: 'Structured Analysis of Parallel Programs' with a subject-oriented modelling language named 'Parallel Activities Specification Scheme'. We use invariably the today apparently prevailing term S-BPM to refer to Fleischmann's approach.

We assume the reader to have some knowledge of the basic concepts of at least one of the S-BPM [32] or the ASM methods [26].

2 Common Features of S-BPM and the ASM Method

The S-BPM and ASM methods share their main goal, namely to reliably link the human understanding of real-life processes to their execution by machines via some implementing software. In fact the ASM method is introduced in [26, p.1] by stating that

> 'The method bridges the gap between the human understanding and formulation of real-world problems and the deployment of their algorithmic solutions by code-executing machines on changing platforms.'

Similarly, a recent presentation of the S-BPM approach states for the 'transformation process of model descriptions to executable ones' [33, Sect.2, p.3-4] that:

> 'end-to-end control is what business stakeholders need to build process-managed enterprise' and that
> 'Any mapping scheme should allow propagating the information from a value chain perspective to a software-development perspective in a coherent and consistent way'.

We explain in this section that as a consequence both methods share three major methodological concerns for descriptions of (concurrent) processes:

- the ground model concern (Sect. 2.1),
- the refinement concern (Sect. 2.2),
- the subject-orientation concern to make the executing agents and their distinct internal and external (communication) actions explicit (Sect. 2.3).

[2] In [15] we showed that the OMG standard BPMN [48], the workflow patterns of the Workflow Pattern Initiative [61] and their (academic) reference implementation YAWL [59] fail to achieve this.

Also both come with 'a *simple scientific foundation*, which adds precision to the method's practicality [26, p.1]'.

Although the two methods realize these three concerns differently, due to the more focussed BPM target of the (current incarnation of the) S-BPM method and the different definitions in the two methods of what constitutes agent behavior (described by Subject Behavior Diagrams (SBDs) resp. ASMs, see Sect. 2.3), and although their scientific foundation comes from different sources, the similarities of the two approaches to software engineering are remarkable because 'the *ground model method* for requirements capture, and the *refinement method* for turning ground models by incremental steps into executable code' form together with the concept of ASMs 'the three constituents of the ASM method for system design and analysis' [26, p.13] through which the method

> '*improves current industrial practice* in two directions:
>
> - On the one hand by accurate high-level modeling at the level of abstraction determined by the application domain ...
> - On the other hand by linking the descriptions at the successive stages of the system development cycle in an organic and effectively maintainable chain of rigorous and coherent system models at stepwise refined abstraction levels.' [26, p.1]

2.1 Ground Model Concern

In the S-BPM literature there is no mention of the name 'ground model' (or 'golden model' as they are called in the semiconductor industry [55]) but the ground model concern is present. The ASM ground model method [7,8,10,12,14] is about constructing prior to code development, as specification for the code, models which are

> 'blueprints that describe the required application-content of programs ... in an abstract and precise form' and are 'formulated in terms of the application domain and at a level of detailing that is determined by the application domain' [14, Sect.1].

Thus ground models satisfy needs of different stakeholders, in particular the domain experts and the software designers. First of all the domain experts (e.g. analysts or users of BPs) need ground models for a '*correct development and understanding* by humans of models and their relation to the application view of the to-be-modeled BP' [15, Sect.5].[3] *Correctness* as used here (together with its

[3] The request in [33, Sect.1,p.1] of a minimal 'semantic distance to human understanding' for S-BPM corresponds to the request for satisfactory ground model ASMs of a 'direct', coding-free relation between the basic domain elements (agents, objects, functions, properties, operations) and the corresponding ASM ground model items [8, Sect.6.2]. The ASM ground model method satisfies this request by offering 'The freedom to choose how to represent the basic objects and operations of the sytem under consideration' and by its attention to 'distinguish between concepts (*mathematical modelling*) and notation (*formalization*)' [8, Sect.5].

companion concept *completeness*) is intrinsically not a mathematical notion, but an epistemological relation between a model and the piece of reality the model is intended to capture, a relation the application experts have to understand and only they (not the software technologists) can judge.

But then also the software designers need ground models, namely as a *complete* specification, where the completeness—every behaviorally relevant feature is stated—makes a correct implementation of the specification reliable. The reliability property links these two roles of ground models. It 'means that the appropriateness of the models can be checked by the application domain experts, the persons who are responsible for the requirements, and can be used by the system developers for a stepwise detailing (by provably controllable ASM refinement steps) to executable code.' [22, p.1923]

Therefore an approach for building satisfactory (i.e. correct, complete and consistent) ground models requires to have solved before 'a *language and communication problem* between the software designers and the domain experts or customers ... the parties who prior to coding have to come to a common understanding of "what to build"' [14, Sect.2.1.1]:

> 'The language in which the ground model is formulated must be appropriate to naturally yet accurately express the relevant features of the given application domain and to be easily understandable by the two parties involved.[4] This includes the capability to calibrate the degree of precision of the language to the given problem, so as to support the concentration on domain issues instead of issues of notation.'(ibid.)(See also the 'language conditions for defining ground models' formulated ibid., Sect.2.3.)

To solve this problem S-BPM starts from two observations of language theory [33, Sect.3, p.5]:

- 'When structuring reality, humans use subjects, predicates and objects.'
- 'humans use natural language structures as primary means to ensure mutual understanding'.

Consequently S-BPM aligns BP descriptions to those three constituents of elementary sentences in natural languages and to the coordination role of communication between subjects.[5] To stay close to natural language, where domain experts formulate process requirements, BP descriptions in S-BPM express the behavior of each subject involved in the BP (read: the agents which perform the described behaviors) as a sequence of possibly guarded basic ('internal') computation or ('external') communication actions of the following form (their content is discussed in Sect. 2.3):

[4] The S-BPM literature speaks about 'duality of expressiveness' which is needed for the description language [33, Sect.2, p.4].

[5] Notably *communication* and *coordination* appear as two of the seven categories of the *Great Principles of Computing* [28].

```
SbpmAction($Condition, subject, action, object$) =
  if $Condition(subject)$ then $subject$ Performs($action$) on $object$
```

These basic S-BPM actions *mutatis mutandis* correspond to basic ASM transitions, even if the two methods have a different view on what is allowed, in general, to constitute an action and on their parallel resp. sequential execution (see Sect. 2.3 and 3.1). In fact in the S-BPM interpreter the ASM rule Behavior($subj, state$)—which formalizes the execution by the *subj*ect of the action (called $service(state)$) associated with its SID-*state*—has exactly the above form, as the reader can check in [32, p.351].

In this way in S-BPM BPs are modeled using a precise language which is understandable by both parties, domain experts (analysts/managers/users) and software developers: it is constituted by elementary sentences which can be understood as (not formalized) natural language sentences, but nevertheless have a precise *operational* meaning (*modulo* a precise meaning of the constituent parts). The resulting BP ground models are as close to the intended real-world processes (read: their intuitive application-domain-views) as are the subjects, their actions and the objects which are chosen by the analyst (as BP model designers are called) to appear in the ground models. Thus the S-BPM approach offers for BPs an interesting solution to a challenge listed in [22, p.1924], namely 'supporting the extraction of ground model elements from natural language descriptions of requirements'.

The 'abstract operational' character of ASM ground models, which makes them directly executable, mentally by definition as well as mechanically by appropriate execution engines, has been recognized in [8, Sect.7] as crucial for the needed '*experimental validation* of the application-domain-based semantical correctness for *ground models*' [14, p.226]. It is a key criterion also for S-BPM, expressed as follows in [33, Sect.1, p.2]:

> 'The novelty of the approach can be summarized by two key benefits, resulting for stakeholders and organization developers:
>
> 1. Stakeholders need only to be familiar with natural language ... to express their work behavior ...[6]
> 2. Stakeholder specifications can be processed directly *without further transformations*, and thus, experienced as described'.

The *ASM ground model method* realizes the ground model concern in a similar way, but tailored for a more general system engineering setting, using the more comprehensive notion of ASM compared to S-BPM's SBDs as they are used to describe the behavior of BP subjects, see below. Not to repeat for an explanation of this difference what has been described in various articles on the theme [7,8,10,12,14] we invite the reader to read the systematic epistemological discussion of the method in [14]. We limit ourselves here to point to a typical

[6] Obviously such a natural language expression of the work behavior has to be sufficiently precise, in particular to avoid misunderstandings that may arise from cultural differences among the stakeholders.

ASM ground model 'at work' S-BPM experts may be interested in, namely the interpreter model for SBDs in [32, Ch.12 and Appendix] (see also [63]). It illustrates the characteristic properties of ASM ground models by exhibiting the direct, strikingly simple and easy to grasp correspondence between the S-BPM concepts and their mathematical, operational formalization by ASMs.

Scientific Foundation. The just mentioned ASM ground model for an SBD-interpreter constitutes the mathematical part of the scientific foundation of S-BPM. The epistemological part of its foundation is rooted in language theory. The ASM method has its simple scientific foundation directly in mathematical logic and its epistemological roots in a generalized Church-Turing thesis (see Sect. 2.3).

2.2 Refinement Concern

In S-BPM the specification of the processes which constitute a BP model is done in two steps. For each process its SBD (also called PASS graph) describes only the sequence in which the executing subject performs its basic actions. The detailed content of these actions is specified by refinements which describe 'the local variables of a process and the operations and functions defined on the local variables' [30, p.206].

Four types of operations and functions are considered, reflecting the classification of actions described in more detail in Sect. 2.3. Two types of *communication* are specified by describing a) the parameters of the communicated messages and b):

- for to-be-received messages the state change they yield, i.e. their 'effect ... on the values of the local variables, depending on the values of the message parameters and the current values of the loca variables' (ibid.)[7]
- for to-be-sent messages the definition of their content depending on the current state, i.e. 'how the values of the message parameters are obtained from the values of the local variables' (ibid.)[8]

So-called *internal operations* are specified by describing their update effect on the current state (here the values of the local variables), where one is allowed to use so-called *internal functions* (whose applications in the current version of S-BPM are not distinguished any more as separate kind of operations), that is mathematical (side-effect-free), in ASM terminology dynamic functions (i.e. functions whose result for given arguments depends on the current state).

To define these specifications and their implementation in S-BPM the approach 'is open for the integration of existing and proved development methods' [30, p.199] and in particular 'all the object oriented concepts can be applied' (ibid., p.206). These two programming-practice inspired refinement types

[7] This is described in the S-BPM interpreter model by the RECORDLOCALLY submachine of ASYNC(*Receive*) and SYNC(*Receive*) [32, p.367-368].

[8] This is described in the S-BPM interpreter model by the functions *composeMsg* and *msgData* of the PREPAREMSG_{Send} submachine [32, p.361].

in S-BPM (Pass graph refinement and its implementation) are instances of the concept of ASM refinement.

The ASM refinement method was conceived in the context of modelling the semantics of ISO Prolog by ASMs [4,5,6,17] (surveyed in [7]), when I was challenged by Michael Hanus to also develop an ASM for the Warren Abstract Machine (WAM)—an early virtual machine whose optimization techniques changed the performance of Prolog to a degree that made practical applications feasible—and to prove the compilation of ISO Prolog to WAM code to be correct. The challenge was solved by refining the Prolog interpreter model in 12 proven to be correct refinement steps to a WAM interpreter model [23,24,25]. The adopted refinement concept (which has been implemented in KIV for a machine verification of the WAM correctness proof [53,54,50,51,52]) is described in detail in [13]. It

- supports sequences of refinement steps whose length depends on the complexity of the to be described system, and
- links the refinement steps in a documented and precise way so that their correctness can be objectively verified.[9]

Since the ASM refinement notion is in essence more general than the programming-focussed one used in S-BPM, we discuss the details in Sect. 3.2.

2.3 Subject-Orientation Concern

In this section we elucidate for the S-BPM and ASM methods the feature which gave the name to S-BPM and is emphasized in the comparative analysis in [30, Ch.5], [32, Ch.14],[33, Sect.4] as distinctive with respect to traditional system description methods, namely the primary role of agents (called subjects) which execute step by step two distinct kinds of actions following the 'program' (behavioral description) each agent is associated with: communications ('external' actions) and 'internal' actions on corresponding objects.

Agents. *Subjects* are placed into the center of S-BPM process descriptions as the 'active elements' of a process which 'execute functions offered by the passive elements' (i.e. objects of abstract data types) [30, p.199] and have to be identified as first elements of any process description: 'start with identifying the involved subjects and after that define the behaviour specifications of acting parties' [33, Sect.3, p.8]. The ASM method shares this view: in the list of the six 'Fundamental Questions to be Asked' when during requirements capture one starts to construct an ASM ground model the first question is:

Who are the system *agents* and what are their relations? [26, p.88]

This corresponds to the fact that by its very definition an ASM is a family of pairs $(a, Pgm(a))$ of different agents, belonging to a set (that may change at runtime), and the (possibly dynamically associated) programs $Pgm(a)$ each

[9] It is an important aspect for certifiability that these verifications are documented to become repeatable by mathematical 'experiment' (read: proof checking). See Sect. 3.3.

agent executes [26, Def.6.1.1].[10] S-BPM has the same definition: 'An S-BPM *process* ... is defined by a set of subjects each equipped with a diagram, called the *subject behavior diagram* (SBD) and describing the behavior of its subject in the process.' [32, p.348] In both definitions we see multiple agents whose behavior is to execute the (sequential) program currently associated with them. Since this happens in a concurrent context, S-BPM and the ASM method both classify the basic 'actions' an agent can perform in a program step by their role for information exchange among the agents, as we are going to explain now.

Classification of Agent Actions. In S-BPM the 'actions' agents perform when executing their program are of two kinds, to 'exchange information and invoke operations' [30, p.372]. Information exchange is understood as sending or receiving messages. The information exchange actions are named 'external' because they involve besides the executing subject also other, 'external' subjects. The invoked other operations are understood as agent-'internal' (read: communication-free) computations on given objects [30, p.205].

Similarly the ASM method explicitly separates agent-internal operations from external data exchange operations (communication) with other cooperating agents, namely through the so-called classification of locations (i.e. containers of abstract data). Agent-internal operations come in the form of read/writes of so-called *controlled* locations which are performed under the complete and exclusive control of the executing agent. Data exchange (communication with cooperating agents) comes in the two forms of a) reading so-called *monitored* locations that are written by the cooperating agents (an abstract form of receiving messages sent by other agents) and b) writing so-called *output* locations to be read by the cooperating agents (an abstract form of sending messages to other agents).

In the interaction view of an S-BPM subject behavior diagram each internal or communication action counts as one step of the corresponding *subject*, namely to perform what is called the *service* associated with the subject in the given state. In the detailed (refined) interpreter view of the *subject* as defined in [32, Appendix, Sect.3] this 'abstract' interaction-view-step usually is rather complex since it is constituted by the sequence of 'detailed-view-steps' performed by the *subject* to execute the underlying internal or communication action— more precisely in the S-BPM interpreter it is the sequence of the START and all PERFORM steps made by the *subject* to execute its BEHAVIOR(*subject*, *SID_state*), otherwise stated the sequence of detailed steps *subject* performs from the moment when it enters the *SID_state* corresponding to the action (read: the associated *service*) until the moment when it exits that state to enter the SID_state' corresponding to the next action, see [32, p.351].

The ASM method started out to provide in full generality the means to abstract into one single-agent step an entire internal computation which may be needed to perform an action in a given state. Therefore one has to separately describe the interaction the considered agent may have with the cooperating agents

[10] To name the agent can be omitted (only) in the special case where a single ASM is contemplated (which may interact with an environment that is considered as run by one other agent).

in its environment to perform the action, namely receiving data from cooperating agents before it starts the abstract step and sending data to cooperating agents after (probably as a result of) the abstract step. The agent's sending interactions are collectively incorporated into its one abstract step, namely as updates of all corresponding output locations; this is without loss of generality given the parallel nature of a single ASM step which performs simultaneously an entire set of location updates. Analogously the agent's receiving interactions directly preceding (and probably influencing) its abstract step are collectively described by a separate so-called 'environment' step which precedes the agent's abstract step and is assumed to be executed by another agent representing the environment of the considered agent; this environment step performs simultaneously all the relevant updates of the corresponding monitored locations, thus completing the definition of the state in which the considered agent performs (the internal part of) its abstract step (see the formal definition in [26, Def.2.4.22, p.75]).

The difference in the technical S-BPM/ASM realization of the identical concept of distinguishing internal and external 'actions' is a result of the different origins of the two methods. The motivating target of S-BPM was to incorporate in an explicit and practically feasible way into the software engineering techniques of the time the missing high-level concept of communication between process agents, in particular for developing BPs where communication is fundamental to control the actions of the cooperating agents. Therefore it was natural to develop an orthogonal communication concept (inspired by CCS [47] and CSP [42]) which is compatible with the principal (at the time prevailingly object-oriented) programming concepts and their implementation so that it can be integrated in a modular way into any practical software engineering method. This led to the interesting input-pool-based S-BPM notion of a synchronous or asynchronous communication (send or receive) 'step' as *pendant* to and *à la pari* with any internal computation 'step'. The notion of an ASM the development of the ASM method started from grew out of an epistemological concern, namely to sharpen the Church-Turing thesis for 'an alternative computation model which explicitly recognizes finiteness of computers' [38,39] (see [11],[26, Ch.9] for the historical details). Therefore it was natural to abstract for the definition of what constitutes an ASM step from any particular form of communication mechanism and to represent a communication (receive or send) action abstractly the same way as any other basic computational action, namely as reading the value of an abstract 'memory location' resp. as updating (writing) it—clearly at the price of having to define an appropriate practical communication model where needed, a task Fleischmann accomplished for S-BPM with his input-pool concept. This concept provides an interesting contribution to the challenge listed in [22, p.1923] to develop 'practically useful patterns for communication and synchronization of multi-agent ASMs, in particular supporting omnipresent calling structures (like RPC, RMI and related middleware constructs) and web service interaction patterns.'[11]

[11] The various theoretical communication concepts surveyed in [41] appear to have been defined to suit parallel and so-called interactive forms of the ASM thesis and seem to have had no practical impact.

Behavior of Agents. In S-BPM the behavior of a single agent is represented by a graph of the Finite State Machine (FSM) flowchart type (called SBD or PASS graph) which 'describes the sequences in which a process sends messages, receives messages and executes functions and operations' [30, p.207]. This corresponds exactly to the so-called control-state ASMs [26, Sect.2.2.6] and their FSM-flowchart like graphical display[12] so that not surprisingly the high-level S-BPM interpreter in [32, Appendix, Sect.7] for the execution of SBDs is defined as a control-state ASM.

3 Differences between S-BPM and the ASM Method

In this section we discuss three major differences between the S-BPM and the ASM method. They concern the notion of *state* and state *change* (update) by actions of agents (Sect. 3.1), the notion of *refinement* of models (Sect. 3.2) and the *verification* concern which helps in the ASM method to increase the system reliability and to reduce the amount of experimental system validations (Sect. 4). Through these features the ASM method offers the practitioner additional possibilities for certifiably correct design of software-intensive systems, although we see no reason why they could not be included into S-BPM, as we are going to suggest, to increase the degree of reliability of S-BPM-designed BPs by certifiable correctness.

3.1 Notion of State and State Change

State. As we have seen in Sect. 2.2, S-BPM shares the traditional programming view of states: 'the values of all local variables define ... the local state of a process' [30, p.206]. In contrast, 'the notion of ASM *states* is the classical notion of mathematical *structures* where data come as abstract objects, i.e. as elements of sets (also called *domains* or *universes*, one for each category of data) which are equipped with basic operations (partial *functions* in the mathematical sense) and predicates (attributes or relations).'[26, p.29] In logic these structures, which have been formulated as a concept by Tarski [58] to define the semantics of first order logic formulae, are also called Tarski structures.[13] The relevant fact for the modelling activity is that the sets and functions which form the state of an ASM can be chosen in direct correspondence with the to-be-modelled items of the application domain, tailored with 'the greatest possible *freedom of language*' [8, Sect. 5] to the intended level of abstraction of the model and 'avoiding the formal

[12] Control-state ASMs have been introduced in [10] as 'a particularly frequent class of ASMs which represent a normal form for UML activity diagrams and allow the designer to define machines which below the main control structure of finite state machines provide synchronous parallelism and the possibility of manipulating data structures.' [26, p.44]

[13] If predicates are considered to be canonically represented by their characteristic functions, a Tarski structure becomes what is called an algebra. Viewed this way an ASM state is a set of functions or Parnas tables [49,9].

system straitjacket' (ibid.). Thus ASM states realize an advice from a great authority: 'Data in the first instance represent abstractions of real phenomena and are preferably formulated as abstract structures not necessarily realized in common programming languages.' [62, p.10]

To provide a characteristic example we can refer to the abstract elements and functions which appear in the ASM model for S-BPM [32, Appendix] as part of the interpreter state, like all the SBD-graph structure related items, the *service*s associated with SID-states and their completion predicate *Completed*, *inputPool* with its related functions, the different sets providing *Alternative*s together with their *select*ion functions, message related functions to *composeMsg* from *msgData*, etc.

Also the object oriented slightly more complex version of the programming view of states as defined above, which comes with the suggestion to use object oriented techniques for the specification of PASS graph refinements [30, p.210], is an instance of the ASM notion of state since 'the instantiation of a relation or function to an object o can be described by the process of parameterization of, say, f to the function $o.f$, which to each x assigns the value $f(o, x)$.'[26, p.29][14]

State Change. The most general kind of a basic action to change a structure or algebra (i.e. a set of functions) appears to be that of a *function update*, i.e. change the value of a function at given arguments, which has the following form:

$$f(t_1, \ldots, t_n) := t$$

Such updates, executed by an agent (denoted by **self**) under appropriate conditions which guard the application of ASM rules:

$$\textsc{AsmRule}_{\mathbf{self}}(\mathit{Condition}, \mathit{Updates}) = \mathbf{if}\ \mathit{Condition}\ \mathbf{then}\ \mathit{Updates}$$

are exactly what constitutes the basic action of an ASM agent in a *state*, where f is an arbitrary n-ary function symbol[15] and $t_1, \ldots, t_n$ are arbitrary terms (expressions) at whose values in the current *state* the new value of the function (which will be the value of the successor state of the current *state*) is set to the value of t in the current *state* (if the indicated condition under which this action is requested to be performed is true in the current state). Given the abstract nature of the functions and objects (elements of the universe) which constitute an ASM state one can express updates at any level of abstraction, using corresponding functions f and expressions t_i, t of given complexity or level of abstraction.

This lifts variable assignment to *destructive assignment at any level of abstraction* and thus supports abstract operational modelling (providing what is nowadays often called execution semantics of a system). A typical use is illustrated by the abstract yet precise definition of the two communication actions

[14] Recently this parameterization facility for ASM states has been exploited to define a general ambient concept in terms of ASMs [16].

[15] 0-ary functions f, i.e. where $n = 0$, are the variables of programming.

$ComAct \in \{Send, Receive\}$ of S-BPM agents by the interpreter submachines Async($ComAct$) and Sync($ComAct$) in [32, Appendix,3.3.,3.4].

Expressivity Question. Due to its original epistemological goal the definition of ASMs had to solve an expressivity issue for the proposed simple algorithmic language, namely to guarantee that this language provides whatever may be needed to 'directly' (coding-free and thus without extraneous overhead) model any computational system. This is what the ASM thesis [38,39] was about and explains why a) the states of ASMs have to be Tarski structures and why b) differently from their static nature in mathematics and logic here these structures must be treated as updatable by basic actions of ASM agents, namely by (a set of simultaneous)[16] updates.

By its focus on modelling BPs by sets of SBDs each of which is described by constructs that are close to sentences of natural language, S-BPM derives the guarantee to be expressive enough for modelling any desired BP from the expressivity of natural language. The price paid is the focus of ground models on the level of abstraction of (sets of) SBDs which are reached by system decomposition (using data flow diagram techniques) until every communicating subject has become explicit,[17] as will become clearer in the next section where we compare the programming-oriented S-BPM refinement concept explained in Sect. 2.2 with the more general ASM refinement notion.

A positive return is the ease with which an S-BPM model can be transformed into a precise (though verbose) natural language text, essentially by paraphrasing each SbpmAction in every SBD of the model by the obvious corresponding natural language sentence. Given the similarity between ASM rules and SbpmActions, in a similar way such a transformation can also be defined for ASM models, as has been illustrated in [20]. There the contributing authors of the book [34] had been asked to formulate in natural language a precise and complete set of requirements for a small case study by first defining a formal specification which captures the given informal requirements and then retranslating this specification into natural language. For S-BPM a converter has been written which transforms S-BPM models into natural language texts [31] (see also [56]). Although we believe that the methodological better way to explain and document ASMs (and also S-SBM models) is to use a *literate modeling* style in the spirit of Knuth's literate programming [45], it could nevertheless be useful

[16] The synchronous parallelism of single-agent actions in the ASM-computation model, which differs from the sequential-program view of actions of S-BPM agents, provides 'a rather useful instrument for high-level design to *locally describe a global state change*, namely as obtained in one step through executing a set of updates' and 'a convenient way to *abstract from sequentiality* where it is irrelevant for an intended design' [8, p.30].

[17] This interesting termination criterion for the 'decomposition of a system into processes'—the first of the two major system development steps in the S-BPM method—is a consequence of the communication focus (read: subject orientation): 'Finally all processes and shared objects, the messages exchanged between processes and the shared objects they use, are identified.' [30, p.204 and Ch.10]

to write a similar *Asm2NatLang* converter to facilitate the integration of ASMs into natural language S-BPM documents for users who are not familiar with symbolic mathematical notations.

3.2 Refinement Concept

The conceptual distance between an SBD (PASS graph) to its refinement, which represents an operational specification of the communication and internal actions the subject performs in the SBD, is not very large. The next step (which we consider as another refinement step) consists in the coding of this specification where the S-BPM method adopts 'methods which are common in standard sequential programming' [30, p.296]. Therefore alltogether the 'semantic gap' between a user model (ground model PASS graph) for a BP and its code is judged not to be very large. In fact it is claimed that 'Once the interaction patterns among actors (subjects) have been refined in terms of exchange of messages, suitable program code can be generated automatically' [33, Sect.1, p.2]; this has to be understood *cum grano salis*, probably meant to hold for 'the standard part of the code' [30, p.295] resp. for code meaning method headers.

This does not solve the problem in case the distance between a ground model and the code is too large to be bridged in one or two steps in such a way that a human can understand the refinement and verify its correctness. Such a situation was at the origin of the ASM refinement method [13] and is typical for its successful applications. Mentioning a few examples should suffice here to illustrate the practical relevance of the ASM refinement notion.

The historically first example is the Prolog-to-WAM compiler verification mentioned in Sect. 2.2 where we needed 12 refinement steps to explain Warren's ideas and to prove the main theorem. The refinement correctness proofs have later been machine verified using the KIV system [53,54]. Interestingly enough to enable the KIV machine to finish its proof, for one of the optimizations in the WAM an additional refinement step had to be introduced into the handwritten proof developed to convince ourselves and our peers. The elaboration of the method for the Occam/Transputer parallel computation model (with non-determinism) yielded 17 natural refinement steps [18] to explain the rationale and prove the correctness of the standard (INMOS) compilation scheme.

Another real-life example to be mentioned (among many others concerning architectures, control software, protocols, algorithms, etc. and surveyed in [26, Ch.9]) is the stepwise refinement of ASM interpreters for Java and the JVM, using both horizontal and vertical refinement steps. These models have been used to verify various properties of interest for the language and its virtual machine, like type safety, compiler correctness, soundness and correctness of the bytecode verifier, soundness of thread synchronization, etc. The reader can find the details in the JBook [57]. That the method could be applied also to C# [19] and .NET CLR [35,37,36] should not come as a surprise.

A natural place to integrate into S-BPM the ASM refinement method is where one has to code complex internal actions of a subject. It is still a challenge to provide tool support for the ASM refinement method, in particular in combination with verifications of refinement correctness, e.g. building upon the implementation of the ASM refinement concept in [50] which has later been extended and been used for numerous other verification projects, see www.informatik.uni-augsburg.de/swt/projects/. Some first steps in this direction seem to appear in the area of software product lines where feature-based modeling is linked to the stepwise validation and verification of properties [60,44,3,27].

3.3 Verification Concern

The presentation of the ASM method quoted at the beginning of Sect. 2 continues as follows:

> 'It covers within a single conceptual framework both *design and analysis*, for procedural single-agent and for asynchronous multiple-agent distributed systems. The means of analysis comprise as methods to support and justify the reliability of software both *verification*, by reasoning techniques, and experimental *validation*, through simulation and testing.' [26, 1]

This shows how much the ASM method cares about both, verification by proving model properties and validation by simulation and testing of models. However it turned out to be an advantage for their use in systems engineering to pragmatically separate these two activites from the modeling (design) activity [8, Sect.4,5], differently from what do other methods (notably the conceptually very close B-method [1,2]) which link design and verification (definition and proof) to always go together.

The ASM method allows one to validate and/or verify properties of models at any level of abstraction since by their definition

- ASMs are mathematical objects so that they satisfy the rigour needed to enter a mathematical or machine supported proof,
- ASMs are conceptually executable, due to their operational character, and have been made mechanically executable by various tools.[18]

Verification cannot replace validation, but as early design-error detection technique it can considerably reduce the amount of testing and error correction after the system is built.

The SAPP/PASS approach shares the validation and verification concern. For 'checking whether a process is correct' two aspects are distinguished [30, p.312, Sect.16.3]:

> - A system must have certain properties, e.g. livelock free, deadlock free which are independent of the application. This is implicit correctness.[19]

[18] See [26, Ch.8] for a survey of various ASM verification and validation tools and [29] for the more recent CoreASM execution engine.

[19] We have pointed out in [15, 4.2] that for BPs 'implicit correctness' properties are less interesting than the ones for 'explicit correctness' which typically are ground model properties to be preserved through refinement steps.

- A specified system must do what a designer has intended. This is explicit correctness.

Both aspects are reported to have been supported by prototypical Prolog-based validation tools providing for each system modeled in PASS a sort of expert system which 'allow(s) the behavior of a process system to be analysed and can determine whether a system does what it was intended to do' (ibid., p.321).

However this verification concern seems not to be supported by the present S-BPM tool set, although the validation concern is, namely by a testing mechanism that allows one to feed concrete values for messages and function arguments and values into the system to run BP scenarios prior to coding method bodies[20].

We suggest to integrate into the current S-BPM system the possibility to

- formulate application-specific BP properties of interest to the user or manager, presumably ground model properties which go beyond the usual graph-theoretic properties like liveness, fairness, deadlock fredom, etc.,
- prove such properties for the ground model as well as their preservation through ASM refinement steps of internal actions,
- document the properties and their verifications so that they can be checked (also by third parties like certification bodies) and used to certify the correctness of the BP implementation.

This could be realized for any of the reasoning techniques the ASM method allows one to apply for the mathematical verification of system properties, at different levels of precision and under various assumptions, e.g. [14, Sect.1]

- outline of a proof idea or *proof sketch* whereby the designers communicate and document their design idea,
- *mathematical proof* in the traditional meaning of the term whereby a design idea can be justified as correct and its rationale be explained in detail,
- *formalized proof* within a particular logic calculus,
- *computer-checked* (automated or interactive) *proof*.

Each technique comes with a different amount of tool support[21] and of effort and cost to be paid for the verification and provides a different level of objective, content-based 'certification' of the professional quality of the analysed system.

4 Evaluation of S-BPM

In this section we evaluate S-BPM as an approach to BPM (Sect. 4.2) using six classical evaluation criteria for practical software engineering methods (Sect. 4.1).

4.1 The Evaluation Criteria

The three major purposes of business process (BP) descriptions are the *design and analysis*, the *implementation* and the *use* of models of BPs. For each purpose

[20] This is exactly the method used in the Falko project at Siemens to validate the ASM ground model for the given scenarios, see [21].

[21] [26, 9.4.3] surveys some tool supported ASM verifications.

pursued by the various BP stakeholders the models play a specific role, namely to serve a) as conceptual models (in particular for high-level development-for-change and management support), b) as specification of software requirements that are implemented by executable models and c) as user model for process execution, monitoring and management. This is reflected in the following six criteria (paraphrased from [15, Sect.5]) a satisfactory BPM system must satisfy:

Ground Model Support. Provide *support for a correct development and understanding* by humans of models and their relation to the application view of the to-be-modeled BP, which is informally described by the process requirements. This human-centered property is often neglected although it is the most critical one for software development systems in general[22] and in particular for BPM systems. It is crucial to support such an understanding *for both model design and use* because these models serve for the communication between

- the BP expert, who has to explain the real-world BP that is to be implemented,
- the IT expert who needs a precise specification of the coding goal,
- the BP user who applies or manages the implemented process and needs to understand for his interaction with the system that his process view corresponds to what the code does.

Refinement Support. Provide *support for faithful implementations* of models via systematic, controlled (experimentally validatable and/or mathematically verifiable) refinements. This model-centered property is methodologically speaking the simpler one to achieve because an enormous wealth of established refinement, transformation and compilation methods can be used for this—if the construction of satisfactory (precise, correct, complete and minimal) ground models is supported the implementation can start from.

Change Management. Provide *support for effective change management* of models. This involves the interaction between machines and humans who have to understand and evaluate machine executions for BP (ground or refined) models, bringing in again conceptual (ground model and refinement) concerns when it comes to adapt the system to evolutionary changes.

Abstraction. Provide *support for abstraction* to help the practitioner in two respects:

- in the daily challenge to develop abstract models (ground models and their stepwise refinements) out of concrete, real-life problem situations. This implies, in particular, the availability in the modeling language of a rich enough set of abstract data types (sets of objects with operations defined on them) to use so that one can
  - express the application-domain phenomena to be modeled (objects and actions) at the conceptual level without the detour of language-dependent encodings;

[22] See the discussion in [14] for the verified software challenge [43] originally proposed by Hoare.

    - refine the abstractions in a controlled manner by more detailed operations on more specific data structures.
  - to develop coherent definitions of different system views (control-flow view, data flow view, communication view, view of the actors, etc.).

Modularization. Provide *support for modularization* through rigorous abstract behavioral interfaces to support structured system compositions into easy-to-change components.[23] For BPM it is particularly important that *modeling-for-change* is supported at all three major stakeholders levels: at the *Ground Model* and *Change Management* support levels because it is the BP users and managers who drive the evolutionary adaptation of BP models, at the *Refinement* support level because the high-level model changes have to be propagated (read: compiled) faithfully to the implementing code.

Practical Foundation. Come with a *precise foundation a practitioner can work with*, i.e. understand and rely upon when questions come up about the behavioral meaning of constructs used by the tool.

4.2 Applying the Criteria to S-BPM

In this section we recapitulate what has been said showing that S-BPM [32] and its tool [46] support correct development and understanding, faithful implementation and effective management of BP models via practical abstraction and modularization mechanisms which are defined on the basis of a fundamental epistemological and mathematically stable foundation.

S-BPM satisfies the *Ground Model* criterion, as shown in Sect. 2.1.

In Sect. 2.2 we have explained to which extent S-BPM satisfies the *Refinement* criterion and in Sect. 3.2 how it can be enhanced to satisfy the full *Refinement* criterion. Modulo the same remark S-BPM satisfies the *Abstraction* criterion.

The *Change Management* criterion is satisfied by S-BPM via its technique to decompose BPs into sets of SBDs, for which in turn modeling for change is supported by two model extension schemes which allow the modeler to smoothly integrate into a given SBD some new (whether normal or interrupt) behavior [32, Appendix, Sect.6].

To satisfy the *Modularization* criterion S-BPM contributes in various ways. Besides the just mentioned constructs for extending normal or interrupt behavior actions can be atomic or composed. In particular structured alternative actions are available. To accurately model alternative (whether asynchronous or synchronous) communication actions it is sufficient to use an appropriate selection function and the traditional iteration construct to loop through the offered alternatives [32, Appendix, Sect.3.1]. For alternative internal actions a structured split-join mechanism is used which allows the modeler to have the selection simply as non-deterministic choice or to condition the choice by static or

[23] These two features, abstraction and modularization, also appear in the *Design* section of *Great Principles Category Narrative* in [28] listed under *simplicity* as one of the five 'driving concerns' of software design and used to 'overcome the apparent complexity of applications'.

dynamic possibly data-related criteria (ibid., Sect.4). Further modular composition constructs include the rigorously defined use of macros, of a normalization to interaction views of SBDs and support for process hierarchies (networks) (ibid., Sect.5).

Notably the model itself which defines the semantics of these features is formulated in a modular way using stepwise ASM refinement (ibid.).

Last but not least S-BPM has a *Practical Foundation* via the accurate definition of its semantics using the language of ASMs—a mathematically precise, wide-spectrum action description language which uses rules of the the same form as guarded basic SBD actions (see Sect. 2.1) and thus is familiar to all BP stakeholders.

References

1. Abrial, J.-R.: The B-Book. Cambridge University Press, Cambridge (1996)
2. Abrial, J.-R.: Modeling in Event-B: System and Software Engineering. Cambridge University Press, Cambridge (2010)
3. Batory, D., Börger, E.: Modularizing theorems for software product lines: The Jbook case study. Universal Computer Science 14(12), 2059–2082 (2008)
4. Börger, E.: A Logical Operational Semantics for Full Prolog. Part I: Selection Core and Control. In: Börger, E., Kleine Büning, H., Richter, M.M., Schönfeld, W. (eds.) CSL 1989. LNCS, vol. 440, pp. 36–64. Springer, Heidelberg (1990)
5. Börger, E.: A Logical Operational Semantics of Full Prolog. Part II: Built-in Predicates for Database Manipulation. In: Rovan, B. (ed.) MFCS 1990. LNCS, vol. 452, pp. 1–14. Springer, Heidelberg (1990)
6. Börger, E.: A Logical Operational Semantics for Full Prolog. Part III: Built-in Predicates for Files, Terms, Arithmetic and Input-Output. In: Moschovakis, Y.N. (ed.) Logic From Computer Science. Berkeley Mathematical Sciences Research Institute Publications, vol. 21, pp. 17–50. Springer, Heidelberg (1992)
7. Börger, E.: Logic programming: The Evolving Algebra approach. In: Pehrson, B., Simon, I. (eds.) IFIP 13th World Computer Congress. Technology/Foundations, vol. I, pp. 391–395. Elsevier, Amsterdam (1994)
8. Börger, E.: Why Use Evolving Algebras for Hardware and Software Engineering? In: Bartosek, M., Staudek, J., Wiedermann, J. (eds.) SOFSEM 1995. LNCS, vol. 1012, pp. 236–271. Springer, Heidelberg (1995)
9. Börger, E.: Evolving Algebras and Parnas tables. In: Ehrig, H., von Henke, F., Meseguer, J., Wirsing, M. (eds.) Specification and Semantics, Schloss Dagstuhl, Int. Conf. and Research Center for Computer Science. Dagstuhl Seminar, No. 9626 (July 1996)
10. Börger, E.: High-level System Design and Analysis using Abstract State Machines. In: Hutter, D., Stephan, W., Traverso, P., Ullmann, M. (eds.) FM-Trends 1998. LNCS, vol. 1641, pp. 1–43. Springer, Heidelberg (1999)
11. Börger, E.: The origins and the development of the ASM method for high-level system design and analysis. Universal Computer Science 8(1), 2–74 (2002)
12. Börger, E.: The ASM Ground Model Method as a Foundation of Requirements Engineering. In: Dershowitz, N. (ed.) Verification (Manna Festschrift). LNCS, vol. 2772, pp. 145–160. Springer, Heidelberg (2004)
13. Börger, E.: The ASM refinement method. Formal Aspects of Computing 15, 237–257 (2003)

14. Börger, E.: Construction and analysis of ground models and their refinements as a foundation for validating computer based systems. Formal Aspects of Computing 19, 225–241 (2007)
15. Börger, E.: Approaches to modeling business processes. A critical analysis of BPMN, workflow patterns and YAWL. Software and Systems Modeling (2011), doi:10.1007/s10270-011-0214-z
16. Börger, E., Cisternino, A., Gervasi, V.: Ambient Abstract State Machines with applications. Computer and System Sciences (2011), Special Issue in honor of Amir Pnueli, http://dx.doi.org/10.1016/j.jcss.2011.08.004
17. Börger, E., Dässler, K.: Prolog: DIN papers for discussion. ISO/IEC JTCI SC22 WG17 Prolog Standardization Document 58, National Physical Laboratory, Middlesex, England (1990)
18. Börger, E., Durdanović, I.: Correctness of compiling Occam to Transputer code. Computer Journal 39(1), 52–92 (1996)
19. Börger, E., Fruja, G., Gervasi, V., Stärk, R.: A high-level modular definition of the semantics of C#. Theoretical Computer Science 336(2-3), 235–284 (2005)
20. Börger, E., Gargantini, A., Riccobene, E.: Abstract State Machines. A method for system specification and analysis. In: Frappier, M., Habrias, H. (eds.) Software Specification Methods: An Overview Using a Case Study, pp. 103–119. HERMES Sc. Publ. (2006)
21. Börger, E., Päppinghaus, P., Schmid, J.: Report on a Practical Application of ASMs in Software Design. In: Gurevich, Y., Kutter, P., Odersky, M., Thiele, L. (eds.) ASM 2000. LNCS, vol. 1912, pp. 361–366. Springer, Heidelberg (2000)
22. Börger, E., Prinz, A.: Quo Vadis Abstract State Machines? Universal Computer Science 14(12), 1921–1928 (2008)
23. Börger, E., Rosenzweig, D.: From Prolog Algebras Towards WAM – a Mathematical Study of Implementation. In: Schönfeld, W., Börger, E., Kleine Büning, H., Richter, M.M. (eds.) CSL 1990. LNCS, vol. 533, pp. 31–66. Springer, Heidelberg (1991)
24. Börger, E., Rosenzweig, D.: WAM Algebras – a Mathematical Study of Implementation, Part 2. In: Voronkov, A. (ed.) RCLP 1990 and RCLP 1991. LNCS (LNAI), vol. 592, pp. 35–54. Springer, Heidelberg (1992)
25. Börger, E., Rosenzweig, D.: The WAM – definition and compiler correctness. In: Beierle, C., Plümer, L. (eds.) Logic Programming: Formal Methods and Practical Applications. Studies in Computer Science and Artificial Intelligence, vol. 11, ch. 2, pp. 20–90. North-Holland (1995)
26. Börger, E., Stärk, R.F.: Abstract State Machines. A Method for High-Level System Design and Analysis. Springer, Heidelberg (2003)
27. Delaware, B., Cook, W., Batory, D.: Product lines of theorems. In: Proc. OOPSLA 2011, Portland (October 2011)
28. Denning, P.J., Martell, C.: Great principles of computing (2007), http://cs.gmu.edu/cne/pjd/GP/GP-site/welcome.html (consulted July 26, 2011)
29. Farahbod, R., et al.: The CoreASM Project, http://www.coreasm.org
30. Fleischmann, A.: Distributed Systems: Software Design and Implementation. Springer, Heidelberg (1994)
31. Fleischmann, A.: Sbpm2NatLang converter. e-mail of September 8 to Egon Börger (2011)
32. Fleischmann, A., Schmidt, W., Stary, C., Obermeier, S., Börger, E.: Subjektorientiertes Prozessmanagement. Hanser-Verlag, München (2011); See [63] for a correct version of the appendix

33. Fleischmann, A., Stary, C.: Whom to talk to? A stakeholder perspective on business process development. Universal Access in the Information Society, pp. 1–26 (June 2011), doi:10.1007/s10209-011-0236-x
34. Frappier, M., Habrias, H. (eds.): Software Specification Methods: An Overview Using a Case Study. HERMES Sc. Publ., Paris (2006)
35. Fruja, N.G.: Type Safety of C# and .NET CLR. PhD thesis, ETH Zürich (2006)
36. Fruja, N.G.: Towards proving type safety of .net cil. Science of Computer Programming 72(3), 176–219 (2008)
37. Fruja, N.G., Börger, E.: Modeling the .NET CLR Exception Handling Mechanism for a Mathematical Analysis. Journal of Object Technology 5(3), 5–34 (2006), `http://www.jot.fm/issues/issue_2006_04/article1`
38. Gurevich, Y.: Reconsidering Turing's Thesis: Toward more realistic semantics of programs. Technical Report CRL-TR-36-84, EECS Department, University of Michigan (September 1984)
39. Gurevich, Y.: A new thesis. Abstracts, American Mathematical Society 6(4), 317 (1985)
40. Gurevich, Y.: Evolving algebras 1993: Lipari Guide. In: Börger, E. (ed.) Specification and Validation Methods, pp. 9–36. Oxford University Press (1995)
41. Gurevich, Y.: Interactive algorithms 2005 with added appendix. In: Goldin, P.W.D., Smolka, S.A. (eds.) Interactive Computation: The New Paradigm, pp. 165–182. Springer, Heidelberg (2006)
42. Hoare, C.A.R.: Communicating Sequential Processes. Prentice Hall (1985)
43. Hoare, C.A.R., Misra, J., Leavens, G.T., Shankar, N.: The verified software initiative: a manifesto. ACM Computing Surveys (2009)
44. Kim, C.H.P., Batory, D., Khurshid, S.: Reducing combinatorics in testing product lines. In: Proc. Aspect Oriented Software Development Conference. ACM (2011)
45. Knuth, D.: Literate Programming. CSLI Lecture Notes, vol. 27. Center for the Study of Language and Information, Stanford (1992)
46. Metasonic. Metasonic-suite, `www.metasonic.de/metasonic-suite`
47. Milner, R.: A Calculus of Communication Systems. LNCS, vol. 92. Springer, Heidelberg (1980)
48. OMG. Business Process Model and Notation (BPMN) (2011), `http://www.omg.org/spec/BPMN/2.0` (formal, January 03, 2011)
49. Parnas, D.L., Madey, J.: Functional documents for computer systems. Sci. of Comput. Program. 25, 41–62 (1995)
50. Schellhorn, G.: Verifikation abstrakter Zustandsmaschinen. PhD thesis, Universität Ulm, Germany (1999)
51. Schellhorn, G.: Verification of ASM refinements using generalized forward simulation. Universal Computer Science 7(11), 952–979 (2001)
52. Schellhorn, G.: ASM refinement and generalizations of forward simulation in data refinement: A comparison. Theoretical Computer Science 336(2-3), 403–436 (2005)
53. Schellhorn, G., Ahrendt, W.: Reasoning about Abstract State Machines: The WAM case study. Universal Computer Science 3(4), 377–413 (1997)
54. Schellhorn, G., Ahrendt, W.: The WAM case study: Verifying compiler correctness for Prolog with KIV. In: Bibel, W., Schmitt, P. (eds.) Automated Deduction – A Basis for Applications, vol. III, pp. 165–194. Kluwer Academic Publishers (1998)
55. Semiconductor Industry Assoc., International technologoy roadmap for semiconductors. Design (2005), `http://www.itrs.net/Links/2005ITRS/Design2005.pdf`
56. Sneed, S.H.: Exporting Natural Language: Generating NL Sentences Out of S-BPM Process Models. In: Fleischmann, A., Schmidt, W., Singer, R., Seese, D. (eds.) S-BPM ONE 2010. CCIS, vol. 138, pp. 163–179. Springer, Heidelberg (2011)

57. Stärk, R.F., Schmid, J., Börger, E.: Java and the Java Virtual Machine: Definition, Verification, Validation. Springer, Heidelberg (2001)
58. Tarski, A.: Der Wahrheitsbegriff in den formalisierten Sprachen. Studia Philosophica 1, 261–405 (1936)
59. ter Hofstede, A., van der Aalst, W., Adams, M., Russell, N. (eds.): Modern Business Process Automation. Springer, Heidelberg (2010)
60. Uzuncaova, E., Khurshid, S., Batory, D.: Incremental test generation for software product lines. IEEE Transactions on Software Engineering 36(3), 309–322 (2011)
61. van der Aalst, W., ter Hofstede, A.: Workflow patterns home page, created and maintained since (1999), http://www.workflowpatterns.com/
62. Wirth, N.: Algorithms & Data Structures. Prentice-Hall (1975)
63. Here the file for the correct text of the appendix of [32] can be downloaded, http://www.hanser.de/buch.asp?isbn=978-3-446-42707-5&area=Wirtschaft, http://www.di.unipi.it/~boerger/Papers/SbpmBookAppendix.pdf

BCNF via Attribute Splitting

Johann A. Makowsky[1] and Elena V. Ravve[2]

[1] Department of Computer Science
Technion - Israel Institute of Technology
Haifa 32000, Israel
janos@cs.technion.ac.il
[2] Department of Software Engineering
ORT Braude College
Karmiel 21982, Israel
cselena@braude.ac.il

Abstract. Boyce-Codd-Heath introduced criteria for good database design, which can be formulated in terms of FD's only. Classical design decomposes relations iteratively using projections. BCNF can not be always achieved using projections alone. 3NF was introduced as a compromise. In this paper we summarize all the known characterizations of BCNF and formulate a new one. In [MR96], attribute splitting was suggested as a heuristics to achieve BCNF in case projections do not do the job. Here we show how attribute splitting can be used to restructure a database scheme iteratively such that the result will be in BCNF, is information preserving and preserves the functional dependencies.

For Bernhard Thalheim on his 60th birthday

1 Introduction

Databases are designed in an interactive way between the database designer and his client. Application driven design uses the language of Entity-Relationship modeling, [MR92, Tha00]. Another approach consists in collecting the attributes U of all the application, and requires from the client that he specifies the functional dependencies F holding between those attributes. After several iterations this results in a big relation $R[U]$ and a set F of functional dependencies. In the remaining design process, $R[U]$ is decomposed using criteria such as avoiding (i) null values, (ii) insertion, deletion and modification anomalies, (iii) redundancies, and achieving optimal storage, while keeping U fixed.

Boyce-Codd-Heath Normal Form for relational databases was introduced to formulate all these properties in terms of F and the key dependencies derivable from F, [Cod70, Cod74, Cod72, Hea71]. The cumulative efforts of many researchers are summarized in Theorem 1, to which we add one more characterization in terms of *hidden bijections*. We also note that for certain standardized translations of the Entity-Relationship Model into the relational model, as given in [MR92], the resulting relation schemes are always in BCNF.

The classical approach to design is based on iteratively decomposing $R[U]$ using projection while keeping U fixed and preserving information and the functional dependencies. Unfortunately, BCNF cannot always be achieved in this way. As a compromise 3NF was formulated, which can always be achieved while preserving information and the functional dependencies. However, not all the FD's follow from the key dependencies. In [MR97], an additional way of restructuring a databases was introduced by splitting attributes: The relation scheme is expanded by new attributes whose interpretation is in bijection with previous attributes. The latter can be expressed using functional dependencies between new and old attributes. The expanded relation scheme then is decomposed into BCNF, preserving information and dependencies up to renaming. In [MR97] this was presented as a heuristics. The suggested procedure always produces relations in BCNF, if it terminates. It is a heuristics, because we could not guarantee termination of the procedure. In this paper we also show termination.

The idea of adding, rather than splitting attributes to achieve relation schemes in BCNF was already discussed in [Sci86]. E. Sciore suggests that if a relation scheme $(R[U], F)$ is not in BCNF, then it is under specified, and its specification can be completed by suitably adding functional dependencies. He then proposes a heuristics how this should be done.

Design theory for relational databases fell out of fashion twenty years ago and few papers were published on the topic. Notable exceptions are [Bis95a, Bis98, VS93, Vin94, Vin98, VL97, FHK+11]. However, with the fashionable trend of XML-driven databases, renewed interest in normal forms emerged, [AL04, AL05, Sch05, MWV04, Tri09]. Hopefully, our approach can be used to formulate design criteria for XML based databases.

We assume the reader is familiar with database theory as described in [AHV94, LL99].

2 BCNF Reviewed

2.1 Functional Dependencies and Normal Forms

Let $U = \{A_1, A_2, \ldots, A_m\}$ be a set of attributes, $R[U]$ a relation scheme over U, and F a set of functional dependencies for $R[U]$ of the form $X \to Y$ with $X, Y \subseteq U$. For simplicity we assume that all the attributes in U are different from each other.

- A functional dependency $X \to Y$ is **trivial** if $Y \subseteq X$.
- F^+ the **deductive closure of** F (with respect to the Armstrong axioms).
- $K \subseteq U$ is a **superkey** for F if $K \to U \in F^+$.
- $K \subseteq U$ is a **(candidate) key** for F if K is a superkey, but no $K' \subset K$ is a superkey.
- The set of **key dependencies** of F is defined by $F_{key} = \{K \to U \in F^+ : K \text{ is a key } \}$.
- Let F be a set of functional dependencies for $R[U]$ and and let $S[U_1]$ be a relation scheme over $U_1 \subseteq U$. We denote by $F[U_1]$ the set $\{X \to Y : XY \subseteq U_1 \text{ and } X \to Y \in F^+\}$, and call it the **projection of** F **on** U_1.

– We denote by triv(U) the set of **trivial dependencies** over U, i.e., the dependencies of the form $X \to Y$ with $X, Y \subseteq U$ and $Y \subseteq X$.

$(R[U], F)$ is in **Boyce-Codd Normal Form (BCNF)** if $(F_{Key})^+ = F^+$. $(R[U], F)$ is in **Third Normal Form(3NF)** if for every non-trivial $X \to Y \in F^+$ either X is a superkey or $Y \subset K$ for some key K for F, i.e., $K \to U \in F^+$. The latter is called a **BCNF-violation for the key** K.

Example 1
The relation scheme $R[NCSZ]$ with

N (Nation), C (City), S (Street), Z (Zipcode)

and $NCS \to Z, NZ \to C$ is in 3NF but not in BCNF: NCS is the key
$NZ \to C$ is a BCNF-violation (NSZ is another key).

Example 2
The relation scheme $R[NSCAP]$ with

N (Name), S (Street), C (City) A (Areacode), P (Phone number)

and $NSC \to AP$, $SC \to A$, is not in 3NF: NSC is the only key.
$R_1[NSCP]$ with $NSC \to P$, and $R_2[SCA]$ with $SC \to A$, are both in BCNF.

2.2 Decompositions of Relation Schemes

Let $R[U]$ be a relation scheme, F a set of functional dependencies.

$(R[U_1], F_1)$ and $(R[U_2], F_2)$ is a **weak decomposition of** $(R[U], F)$ if $U_1 \cup U_2 = U$, and furthermore ${F_i}^+ \subset F[U_i]^+$. It is a **strong decomposition** if additionally ${F_i}^+ = F[U_i]^+$.

The decomposition $(R[U_1], F_1)$ and $(R[U_2], F_2)$ is **information preserving**, if for all instances r of R satisfying F we have that $r = \pi_{U_1}(r) \bowtie \pi_{U_2}(r)$.

It is **FD preserving**, if $(F_1 \cup F_2)^+ = F^+$.

Remark 1. If $(R[U_1], F_1)$ and $(R[U_2], F_2)$ is an information and dependency preserving decomposition of $(R[U], F)$, $A \in U_1 \cap U_2$, and $r_1 \models F_1$ and $r_2 \models F_2$, it is not required that $\pi_A r_1 = \pi_A r_2$. This is only the case for an instance $r \models F$ and $r_i = \pi_{U_i} r$.

In the literature, decompositions require $F_i = F[U_i]$, which is equivalent to $(F_i)^+ = F[U_i]$. In our terminology they are strong decompositions.

The relation scheme $R[ABCD]$ with $F = \{C \to D, B \to A, CD \to B\}$ can be decomposed into $(R[CD], F_1)$ with $F_1 = \{C \to D\}$, $(R[BCD], F_2)$ with $F_2 = \{CD \to B\}$ and $(R[AB], F_3)$ with $F_3 = \{B \to A\}$. Note that $C \to D \notin F_2$ but $C \to D \in F[BCD]$, so this is not a strong decomposition, but it preserves information and dependencies.

We will make use of our relaxed notion of weak decomposition in Section 3.

2.3 Redundancy

Let (R, F) be a relation scheme. R is **F-redundant (F^+-redundant) on XY** if there exists a relation $r \models F$ and a non-trivial FD $X \rightarrow Y \in F$ ($\in F^+$), and at least two distinct tuples $t_1, t_2 \in r$ with $t_1[XY] = t_2[XY]$. R is **F-redundant (F^+-redundant)** if there is $XY \subset U$ such that R is F-redundant (F^+-redundant) on XY.

We distinguish sets of attributes of the form XY as follows: If $X \rightarrow Y \in F$ and not trivial, XY is called an **explicit fact**. If $X \rightarrow Y \in F^+ - F$ and not trivial, XY is called an **implicit fact**.

Now, $R[U]$ is F-redundant (F^+-redundant) on $XY \subset U$ iff XY is a fact and XY is not a superkey. The rationale behind redundancy is, that if R is redundant on an explicit or implicit fact XY, the fact should be stored in a different table. R is **not** F-redundant (F^+-redundant) if every fact is a superkey.

2.4 Anomalies

Anomalies were introduced in [Fag79]. We are given a relation scheme $R[U]$ and a set of FD's F with a set of candidate keys given by F_{Key}. Let r be a relation for R with $r \models F$. Let $t[U]$ be a tuple we want to insert. We check whether $r \cup \{t[U]\} \models F_{Key}$. If $r \cup \{t[U]\} \models F_{Key}$ we accept, else we reject the insertion of $t[U]$. If we accept, but $r \cup \{t[U]\} \not\models F$, we say that $t[U]$ is an **insertion violation**.

R, F has an **insertion anomaly** if there is an r and $t[U]$, which is an insertion violation.

Let $t[U] \in r$ be a tuple we want to delete. Again we check whether $r - \{t[U]\} \models F_{Key}$. If $r - \{t[U]\} \models F_{Key}$ we accept, else we reject the deletion of $t[U]$. If we accept, but $r - \{t[U]\} \not\models F$, we say that $t[U]$ is a **deletion violation**.

R, F has a **deletion anomaly** if there is an r and $t[U]$, which is a deletion violation.

However, let $s \subseteq r$ be another relation for R. If $r \models F$, so also $s \models F$. Hence, there are no deletion anomalies for FD's.

Let r be a relation for $R[U], F$, $t \in r$, $r \models F$, K_0 be a fixed candidate key for F. Let t' be a tuple such that $(r - \{t\}) \cup \{t'\} \models F_{Key}$ and one of the following:

(i) $t[K] = t'[K]$ for some candidate key for F;
(ii) $t[K_0] = t'[K_0]$ for the specifically chosen key K_0;
(iii) $t[K] = t'[K]$ for every candidate key for F;

but $(r - \{t\}) \cup \{t'\} \not\models F$. Then r and t' show a **modification anomaly** M_i, M_{ii}, M_{iii} respectively.

Because $F_{key} \models F$, we note that if R, F is in BCNF then it has no modification anomaly M_i (and hence neither M_{ii} and M_{iii}).

2.5 Unpredictable Insertions

Unpredictable insertion were introduced in [BG80]. We use their terminology. Up to logical equivalence within natural language their definition is really the same as in [Fag79]. Let $R[U], F$ be a relation scheme. An insertion of a tuple t into $r \models F$ is said to be **F-valid**, if $r \cup \{t\} \models F$. A set of attributes $X \subseteq U$ is said to be **unaffected** by a valid insertion $r' = r \cup \{t\}$ iff $\pi_X(r) = \pi_X(r')$. A valid insertion is **F-unpredictable** (F^+-unpredictable) if there exists a non-trivial $X \to Y \in F$ ($X \to Y \in F^+$) such that XY is unaffected by it.

2.6 Storage Saving

Let r be an instance of the relation scheme $(R[U], F)$. For $V \subseteq U$ we define

$$n_V(r) = |\{t[V] : t \in r\}|$$

and

$$s_V(r) = |\{t[V] : t \in r\}| \cdot |V|$$

$n_V(r)$ counts the number of tuples in r which are different on V, and $s_V(r)$ is the size (in terms of attribute values) of $\pi_V(r)$.

Let $\pi_{U_i} R = R_i[U_i]$ be a **information preserving decomposition** of $R[U]$ which is **non-trivial**, i.e., $U_i \neq U$. We say that the decomposition is **storage saving** if there are instances $r = \bowtie_i r_i$ such that $\sum_i s_{U_i}(\pi_{U_i} r) \leq s_U(r)$.

Example 3
Consider $R[ABCD]$ with
$F_{3.1} = \{A \to BCD, C \to D\}$ (not in BCNF) and
$F_{3.2} = \{A \to BCD, C \to A\}$ (in BCNF).
We decompose R into $R_1[ABC]$ and $R_2[CD]$ for $F_{3.1}$ and $S_1[AC]$ and $S_2[ABD]$ for $F_{3.2}$. With $F_{3.1}$ there may be fewer values for C than for A, but with $F_{3.2}$ this is not possible.

More generally we have

Lemma 1. *Let $(R[U], F)$ be a relation scheme with $U = KVW$ and K is a key, i.e., $K \to U \in F^+$. Then for all instances $r \models F$ we have*

$$s_{KV}(r) + s_{KW}(r) > s_U(r)$$

and

Lemma 2. *Let $(R[U], F)$ be a relation scheme with $U = KVW$ and K is not a superkey, in particular $K \to V \notin F^+$. Then for every $\epsilon \in \mathbb{R}$ there are instances $r \models F$ such that*

$$\frac{s_{KV}(r) + s_{KW}(r)}{s_U(r)} \leq (1 + \epsilon)\frac{|KW|}{|U|}$$

Proof. For r with N tuples and only one value for K and W we have N values for V. Therefore

$$\frac{s_{KV}(r) + s_{KW}(r)}{s_U(r)} = \frac{N + 1}{N}$$

which converges to 1. On the other hand we have

$$\frac{|KW|}{|U|} = \frac{2}{3},$$

hence the lemma is proven. □

It is now easy to see, as was observed in [Bis95a, Bis98, VS93], that a relation scheme (R, F) is in BCNF iff it has no storage saving decomposition.

2.7 Characterizing BCNF

The following summarizes the classical and less known characterization of BCNF, cf. [Bis95b, LL99] and [Fag79, BG80, Bis95a, Bis98, VS93, Vin94, Vin98].

Theorem 1 (BCNF-characterization Theorem)
Let F be a set of functional dependencies over a relation scheme (R, F). The following are equivalent:

(i) (R, F) *is NOT in BCNF;*
(ii) (R, F) *is redundant with respect to* F*;*
(iii) (R, F) *has an insertion anomaly with respect to* F*;*
(iv) (R, F) *has a modification anomaly with respect to* F*.*
(v) (R, F) *has* F*-unpredictable insertions.*
(vi) (R, F) *has a storage saving information preserving decomposition.*

2.8 Hidden Bijections

Let $U_1, U_2 \subseteq U$ be two sets of attributes with functional dependencies $U_1 \rightarrow U_2$ and $U_2 \rightarrow U_1$. This establishes a bijection between the data in U_1 and U_2, which can be viewed as yet another form of redundancy. We shall give one more characterization of BCNF in terms of the absence of such bijections.

Let $(R[U], F)$ be a relation scheme with $V, X, Y \subseteq U$ sets of attributes and F a set of FD's. We say that F has a **hidden bijection** if $VX \leftrightarrow VY \in F^+$ and at least one $Y \rightarrow X \in F^+ - F^+_{key}$ or $X \rightarrow Y \in F^+ - F^+_{key}$ or both hold. The rôles of X and Y are **not** symmetric. Moreover, if $Y \rightarrow X \in F^+ - F^+_{key}$ then w.l.o.g., we take $X = \{B\}$. In addition, V and Y can be assumed to be minimal and VXY is not necessarily equal to U.

Remark 2. We call this a *hidden* bijection because X and Y are not in bijection, but VX and VY are.

Let us look again at Example 1 $R[NCSZ]$ with

N (Nation), C (City), S (Street), Z (Zipcode)

and $NCS \rightarrow Z, NZ \rightarrow C$. The dependency $NZ \rightarrow C \in F^+ - F^+_{key}$ and the fact that $V = NS$ gives a hidden bijection. We note that $NZ \rightarrow C \in F^+ - F^+_{key}$ is a BCNF-violation.

Theorem 2
$(R[U], F)$ is in BCNF iff it has no hidden bijections.

Proof

- If $(R[U], F)$ is in BCNF then $F^+ - F^+_{key} = \mathrm{triv}(U)$ and there is no candidate to hidden bijections.
- Conversely, if $(R[U], F)$ is not in BCNF then $F^+ - F^+_{key} \neq \mathrm{triv}(U)$ and there exists a non-trivial $Y \rightarrow B \in F^+ - F^+_{key}$. Without loss of generality, we assume that our scheme is in 3NF. It leads to two options:
  - (i) Y is a superkey and there exists a (minimal)[1] key K such that $K \subset Y$. In this case, $Y \rightarrow B \in F^+ - F^+_{key}$ is a hidden bijection with $VY \leftrightarrow VB$, where $V = K$.
  - (ii) $B \in K$, where K is a (minimal) key and $Y \neq K$ is not a superkey. In this case, $Y \rightarrow B \in F^+ - F^+_{key}$ is a hidden bijection with $VY \leftrightarrow VB$, where $V = K - \{B\}$.

□

Remark 3. If we do not request V to be minimal then we take $V = U - \{B\}$. So, we have: $VB \rightarrow Y \in F^+$ as $VB = U$ and $VY \rightarrow B \in F^+$ as $Y \rightarrow B \in F^+$.

3 Attribute Splitting

3.1 Motivating Example

The following analysis of the standard example from [Ull82] of a BCNF violation is slightly modified and taken mostly from [MR96].

Example 1 of $R[NCSZ]$ with N: Nation, C: City, S: Street, Z: Zipcode and $F = \{NCS \rightarrow Z,\ NZ \rightarrow C\}$ is in 3NF but not in BCNF. The only BCNF-violation is $NZ \rightarrow C$. We have added an additional attribute N such that the left hand side of $NZ \rightarrow C$ is not a singleton.

We can modify it to be in BCNF in two ways:

Either we drop the BCNF-violation $NZ \rightarrow C$ altogether. This would make sense if the character of postal distribution has changed and only new zip-codes are introduced.

Otherwise, we split the zip-code, as is done in practice in many countries. This allows us to keep the old data intact and store the modification in additional tables. In further updates these tables do not have to be updated, if the new zip-code is not related to any old zip-codes.

We split Z into Z_{city} and Z_{local} with new FD's $NCS \rightarrow Z_{local}$, $NZ_{city} \rightarrow C$, $NC \rightarrow Z_{city}$ and a new relation scheme $R'[NCSZ_{local}Z_{city}]$.

Each tuple t in an instance r of $R[NCSZ]$ is extended to a tuple $t'[NCSZ_{city}Z_{local}]$ such that $t'[NCS] = t[NCS]$ and $t'[Z_{city}Z_{local}] = t[CZ]$. Because NZ_{city} and NC are in a bijection, we can decompose R' in two ways.

[1] In the standard textbooks there is some confusion: in some books keys are superkeys, in other books keys are always minimal. We added minimal here in parentheses to avoid this ambiguity.

(i) $Address_1[NCSZ_{local}]$ with $NCS \rightarrow Z_{local}$, and
$CityCode[NCZ_{city}]$ with $NZ_{city} \rightarrow C$ and $NC \rightarrow Z_{city}$, or
(ii) $ZipCode[NSZ_{city}Z_{local}]$ with $NSZ_{city} \rightarrow Z_{local}$, and
$CityCode[NCZ_{city}]$ with $NZ_{city} \rightarrow C$ and $NC \rightarrow Z_{city}$.

This gives as two tables instead of one. However, we can gain storage space provided that Z_{City} is a short code for city names, and Z_{local} is a short code for sets of street names. Note that saving storage must be measured in number of attribute values and tuples, not in the number of tuples. If we drop the BCNF-violation from our requirements, we save even more storage. We can use the unused zip-codes resulting from imbalances of city-size. For example, in USA, New York has many zip-codes, say 001-0001 up to 001-9999, whereas Montauk has very few, say 002-0001 up to 002-0009. With $NZ \rightarrow C$ the values 002-0010 up to 002-9999 are waisted. We can also gain by grouping small cities into bigger areas with same first three digits.

In both cases the decomposition is information preserving, and all the relation schemes are in BCNF with respect to their FD's. However, the original zip-code attribute Z is lost. It can be saved by adding a third relation scheme $ZipSplit[ZZ_{city}Z_{local}]$ with FD's $Z \rightarrow Z_{city}Z_{local}$ and $Z_{city}Z_{local} \rightarrow Z$.

Do we preserve all the dependencies of the original design. In case (ii) this is not the case because the attributes of $NCS \rightarrow Z_{local}$ do not appear together in a relation scheme. In case (i) we have to check whether F^+ is contained in

$$\{NCS \rightarrow Z_{local}, NZ_{city} \rightarrow C, NC \rightarrow Z_{city}, Z \rightarrow Z_{city}Z_{local}, Z_{city}Z_{local} \rightarrow Z\}^+$$

The reader can check that this is not the case. In the simpler case where the attribute N is omitted both from the original relation scheme and from all the FD's the FD's are preserved.

In our attribute splitting example the relation schemes $Address_1[NCSZ_{local}]$, $CityCode[NCZ_{city}]$ and $ZipCode[NSZ_{city}Z_{local}]$ are projections of R'. However $ZipSplit[ZZ_{city}Z_{local}]$ is not.

3.2 *AB*-Splitting

For the sequel we have to define **substitution of attributes** in sets of functional dependencies. Let U be a set of attributes, $A \in U$ and $A_{new} \notin U$. For a set F of dependencies over U we denote by $F|^{A}_{A_{new}}$ the set of dependencies in F where each appearance of A is replaced by A_{new}. We note that $(F|^{A}_{A_{new}})|^{A_{new}}_{A} = F$.

Let $(R[U], F)$ be a relation scheme with $Y \subset U$ a set of attributes, $A, B \in U$ attributes and F a set of FD's, and $Y \rightarrow B \in F^+ - F^+_{key}$, while $A \in Y$ and $Y - \{A\} \rightarrow B \notin F^+$, and V is a key, such that $B \in V$.

We want to restructure $(R[U], F)$ in such a way that the BCNF-violation $Y \rightarrow B$ is eliminated without loss of information or dependencies, and without introducing new BCNF-violations.

To achieve this we shall expand $(R[U], F)$ by adding three new attributes A_{new}, A_A, A_B to form $\bar{R}, \bar{F}$ in the following way:

(i) $\bar{F} = F \cup \{AB \rightarrow A_{new}, A_{new} \rightarrow AB, A_A \rightarrow A, A \rightarrow A_A, A_B \rightarrow B, B \rightarrow A_B\}$
(ii) Each instance $r \models F$ is expanded into an instance $\bar{r}$ for $\bar{R}, \bar{F}$ as follows: $t \in r$ is expanded to $\bar{t}$ by setting $\bar{t}[U] = t[U]$ and $\bar{t}[A_A] = t[A]$, $\bar{t}[A_B] = t[B]$ and $\bar{t}[A_{new}] = t[AB]$.

Proposition 1. *(i) Every* $r \models F$ *can be expanded to* $\bar{r} \models \bar{F}$.
(ii) $\pi_U(\bar{r}) = r$ *and* $\pi_U(\bar{r}) \models F$ *iff* $\bar{r} \models \bar{F}$.

Now we decompose $\bar{R}$ into three relation schemes:

(i) $U_1 = U - \{A\}A_{new}$ and $(R_1[U_1], F_1)$ with $F_1 = (F - \{Y \rightarrow B\})^+|^A_{A_{new}})$;
(ii) $U_2 = A_B B$ and $(R_2[U_2], F_2)$ with $F_2 = \{A_B \rightarrow B, B \rightarrow A_B\}$.
(iii) $U_3 = A_{new}A_A A_B$ and $(R_3[U_3], F_3)$ with
$F_3 = \{A_{new} \rightarrow A_A A_B, A_A A_B \rightarrow A_{new}\}$

This is a weak decomposition of $(\bar{R}, \bar{F})$ but not a strong decomposition. Furthermore, as (R, F) was not in BCNF there are instances $\bar{r} \models \bar{F}$ such that

$$s_{U-\{A\},A_{new}}(\bar{r}) + s_{A_B B}(\bar{r}) + s_{A_{new}A_A A_B}(\bar{r}) < s_U(\bar{r}) = s_U(r)$$

In other words expanding (R, F) to $(\bar{R}, \bar{F})$ is still storage saving.

Interpretation: Instead of just forgetting $Y \rightarrow B$, we restructure the attribute $A \in Y$ by replacing it by A_{new} and removing the BCNF-violation from R_1. The new relations (R_2, F_2) and (R_3, F_3) describe the restructuring and preserve $A \in Y$ implicitly in tables R_2 and R_3 which will be shown to be in BCNF.

Proposition 2. *The decomposition of* $(\bar{R}, \bar{F})$ *into* $(R_1, F_1), (R_2, F_2), (R_3, F_3)$ *is information preserving, i.e., for every* $\bar{r} \models \bar{F}$ *we have*

$$\pi_{U-\{A\}A_{new}}\bar{r} \bowtie \pi_{A_B B}\bar{r} \bowtie \pi_{A_{new}A_A A_B}\bar{r} = \pi_{U-\{A\}}\bar{r}$$

Proof. The left join is on B and B is a key of R_2. The right join is on A_{new} and A_{new} is a key of R_3. □

Proposition 3. *Given* (R, F), *let* $(\bar{R}, \bar{F})$ *be the expansion defined above. The decomposition of* $(\bar{R}, \bar{F})$ *into* $(R_1, F_1), (R_2, F_2), (R_3, F_3)$ *preserves dependencies in the following sense:*

$$F^+|^A_{A_{new}} \subseteq (F_1 \cup F_2 \cup F_3)^+[U - \{A\}A_{new}]$$

Proof. We first prove $Y - \{A\}A_{new} \rightarrow B \in (F_2 \cup F_3)^+$.

$A_{new} \rightarrow A_B \in F_3$. $A_B \rightarrow B \in F_2$. By transitivity we get $A_{new} \rightarrow B$. By augmentation we conclude $Y - \{A\}A_{new} \rightarrow B \in (F_2 \cup F_3)^+$.

Next we observe that $(F_1 \cup \{Y - \{A\}A_{new} \rightarrow B\})^+|^{A_{new}}_A = F^+$ or equivalently, $(F_1 \cup \{Y - \{A\}A_{new} \rightarrow B\})^+ = (F^+)|^A_{A_{new}}$. □

Observation 1. (R_2, F_2) *and* (R_3, F_3) *are in BCNF.*

About the status of (R_1, F_1) we can say little in general. However, we observe that,

Proposition 4. *If (R, F) is in 3NF, G is a minimal cover of F, and the dependency $Y \to B \in G$ is a BCNF-violation, then $G - \{Y \to B\}$ is a minimal cover of $F' = F^+ - \{Y \to B\}$ and (R, F') is in 3NF and has fewer BCNF violations than (R, F).*

We can iterate this construction to achieve BCNF in the following way:

(i) Given a relation scheme (R^0, F^0) we can always put it into 3NF using the synthesis algorithm.
(ii) So without loss of generality we can assume that our (R, F) is one of the relation schemes resulting from the synthesis algorithm which is in 3NF but not in BCNF.
(iii) Furthermore, if (R, F) is not in BCNF, it has a minimal cover G of F which contains a BCNF-violation, say $Y \to B$ and $A \in Y$.
(iv) We apply our AB-splitting to eliminate this BCNF violation in (R, F).
(v) Using Propositions 1 and 2 we see that we preserve information and the dependencies.
(vi) Using Proposition 4 we reduce the number of BCNF-violations because G is a minimal cover and $Y \to B \in G$.
(vii) We did not introduce any BCNF-violations which were not in G^+ because we used a weak decomposition where $F_1 \neq F[U_1]$, more precisely the dependency $Y \to B$ is in $F[U_1] - F_1$.

Remark 4. If $Y = \{A\}$ is a singleton, we can simplify AB-splitting as follows: We split A into A_A, A_B and define (R_i, F_i) by

(i) $(R_1[U - \{A\}, A_A], F_1)$ with $F_1 = (F - \{A \to B\})^+|^{A}_{A_A})$,
(ii) $(R_2[A_B, B], F_2)$ with $F_2 = \{A_B \to B, B \to A_B\}$,
(iii) $(R_3[A, A_A, A_B], F_3)$ with $F_3 = \{A \to A_A A_B, A_A A_B \to A\}$,

and $\bar{r}$ is defined as before.

4 Conclusion

We have collected all the known characterizations of BCNF in terms of the underlying functional dependencies, and we have added one more characterization in terms of hidden bijections.

Given $(R[U], F)$, it is not always possible to achieve BCNF using strong decomposition only while keeping U fixed and preserving information and the dependencies. Instead we restructure $(R[U], F)$ by adding attributes to U which are in bijection with some of the original attributes. We then decompose the restructured database scheme into relations schemes, using a weak decomposition $(R[U_i], F_i)$ in such a way that $F_i \subseteq F[U_i]$. In this way we finally get a database scheme in BCNF which preserves information and the dependencies are preserved up to renaming.

Acknowledgments. We would like to thank E. Fischer and Y. Kanza for discussing drafts of this paper with us, and to Y. Kanza for reading carefully the current version and suggesting expository improvements.

References

[AHV94] Abiteboul, S., Hull, R., Vianu, V.: Foundations of Database. Addison Wesley (1994)

[AL04] Arenas, M., Libkin, L.: A normal form for XML documents. ACM Transactions on Database Systems 29(1), 195–232 (2004)

[AL05] Arenas, M., Libkin, L.: An information-theoretic approach to normal forms for relational and XML data. Journal of ACM 52(2), 246–283 (2005)

[BG80] Bernstein, P.A., Goodman, N.: What does Boyce-Codd normal form do? In. In: Sixth Conference on VLDB, pp. 245–259 (1980)

[Bis95a] Biskup, J.: Database Schema Design Theory: Achievements and Challenges. In: Bhalla, S. (ed.) CISMOD 1995. LNCS, vol. 1006, pp. 14–44. Springer, Heidelberg (1995)

[Bis95b] Biskup, J.: Grundlagen von Informationssystemen. Vieweg (1995)

[Bis98] Biskup, J.: Achievements of relational database schema design theory revisited. In: Thalheim, B., Libkin, L. (eds.) Semantics in Databases, pp. 29–54. Springer, Berlin (1998)

[Cod70] Codd, E.F.: A relational model of large shared data banks. Communications of the ACM 13(2), 377–387 (1970)

[Cod72] Codd, E.F.: Further normalization of the data base relational models. In: Rustin, R. (ed.) Data Base Systems, pp. 33–64. Prentice-Hall, Englewood Cliffs (1972)

[Cod74] Codd, E.F.: Recent investigation in relational data base systems. In: IFIP Proceedings, pp. 1017–1021 (1974)

[Fag79] Fagin, R.: Normal forms and relational database operators. In: Proceedings of ACM-SIGMOD Conference on Management of Data, pp. 153–160 (1979)

[FHK$^+$11] Ferrarotti, F., Hartmann, S., Köhler, H., Link, S., Vincent, M.: The Boyce-Codd-Heath Normal Form for SQL. In: Beklemishev, L.D., de Queiroz, R. (eds.) WoLLIC 2011. LNCS, vol. 6642, pp. 110–122. Springer, Heidelberg (2011)

[Hea71] Heath, I.J.: Unacceptable file operations in a relational data base. In: Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access, and Control, San Diego, Ca., pp. 19–33 (1971)

[LL99] Levene, M., Loizou, G.: Guided Tour of Relational Databases and Beyond. Springer, London (1999)

[MR92] Mannila, H., Räihä, K.J.: The Design of Relational Databases. Addison-Wesley (1992)

[MR96] Makowsky, J.A., Ravve, E.: Translation Schemes and the Fundamental Problem of Database Design. In: Thalheim, B. (ed.) ER 1996. LNCS, vol. 1157, pp. 5–26. Springer, Heidelberg (1996)

[MR97] Makowsky, J.A., Ravve, E.V.: Dependency preserving refinement and the fundamental problem of database design. Data and Knowledge Engineering 24(3), 277–312 (1997)

[MWV04] Liu, C., Vincent, M.W., Liu, J.: Strong functional dependencies and their application to normal forms in XML. ACM Transactions on Database Systems 29(3), 445–462 (2004)

[Sch05] Schewe, K.-D.: Redundancy, dependencies and normal forms for xml databases. In: Dobbie, G., Williams, H. (eds.) Sixteenth Australasian Database Conference (ADC 2005). Conferences in Research and Practice in Information Technology, vol. 39, pp. 7–16. University of Newcastle, Newcastle (2005)

[Sci86] Sciore, E.: Comparing the universal instance and relational data models. In: Kanellakis, P.C., Preparata, F. (eds.) The Theory of Databases. Advances in Computing Research, vol. 3, pp. 139–163. JAI Press, Inc., Greenwich (1986)

[Tha00] Thalheim, B.: Entity-Relationship Modeling: Foundations of Database Technology. Springer, Heidelberg (2000)

[Tri09] Trinh, D.-T.: XML Functional Dependencies based on Tree Homomorphisms. PhD thesis, Faculty of Mathematics/Informatics and Mechanical Engineering, Clausthal University of Technology, Clausthal, Germany (2009)

[Ull82] Ullman, J.D.: Principles of Database Systems, 2nd edn. Principles of Computer Science Series. Computer Science Press (1982)

[Vin94] Vincent, M.W.: The Semantic Justification of Normal Forms in Relational database Design. PhD thesis, Department of Computer Science, Monash University, Australia (1994)

[Vin98] Vincent, M.W.: Redundancy Elimination and a New Normal Form for Relational Database Design. In: Thalheim, B., Libkin, L. (eds.) Semantics in Databases 1995. LNCS, vol. 1358, pp. 247–264. Springer, Heidelberg (1998)

[VL97] Vincent, M.W., Levene, M.: Restructuring partitioned normal form relations without information loss. In: Proceedings of International Conference on Management of Data (COMAD), Bombay, pp. 111–124 (1997)

[VS93] Vincent, M.W., Srinivasan, B.: A note on relation schemes which are in 3NF but not in BCNF. Information Processing Letters 48, 281–283 (1993)

Foundations for a Fourth Normal Form over SQL-Like Databases

Flavio Ferrarotti[1], Sven Hartmann[2], Henning Köhler[3], Sebastian Link[4], and Millist W. Vincent[5]

[1] School of Information Management, Victoria University of Wellington, New Zealand
[2] Institut für Informatik, Technische Universität Clausthal, Germany
[3] N-Squared Software, Palmerston North, New Zealand
[4] Department of Computer Science, University of Auckland, New Zealand
[5] School of Computer and Information Science, University of South Australia, Australia

Abstract. In the relational model of data the Fourth Normal Form condition guarantees the elimination of data redundancy in terms of functional and multivalued dependencies. For efficient means of data processing the industry standard SQL permits partial data and duplicate rows of data to occur in database systems. Here, the combined class of uniqueness constraints, functional and multivalued dependencies is more expressive than the class of functional and multivalued dependencies itself. Consequently, the Fourth Normal Form condition is not suitable for SQL databases. We characterize the associated implication problem of the combined class in the presence of NOT NULL constraints axiomatically, algorithmically and logically. Based on these results we are able to establish a suitable Fourth Normal Form condition for SQL.

1 Introduction

In the *relational model of data* [4] a relation schema consists of a finite set R of attributes A that have a countably infinite domain $dom(A)$. A relation over R is a finite set of tuples, i.e. elements of the cartesian product over the domains. Data engineers specify finite sets Σ of integrity constraints over R to restrict the relations to those considered meaningful to the application at hand. A functional dependency (FD) over R is an expression $X \to Y$ with $X, Y \subseteq R$. It restricts relations to those where every pair of tuples with the same values on all the attributes in X also has the same values on all the attributes in Y. A multivalued dependency (MVD) over R is an expression $X \twoheadrightarrow Y$ with $X, Y \subseteq R$. It restricts relations to those where every pair of tuples with the same values on all the attributes in X implies the existence of some tuple in the relation that has the same values on all the attributes in XY as the first tuple, and the same values on all the attributes in $X(R-Y)$ as the second tuple. FDs and MVDs are essential for database design and data processing: if there is an MVD $X \twoheadrightarrow Y$ over R with $X \neq XY \neq R$, then either all the attributes of $R-X$ are functionally dependent on X or there are relations with redundant data value occurrences.

Redundancy can lead to inefficiencies with updates. A relation schema R with a set Σ of FDs and MVDs is in *Fourth normal form* (4NF) [10], if for every MVD $X \twoheadrightarrow Y$ implied by Σ, $Y \subseteq X$ or $XY = R$ or the FD $X \to R$ is implied by Σ.

Example 1. The schema WORK with attributes *Employee*, *SSNo* (Social Security Number), and *PassNo* (Passport Number), is in 4NF with respect to the set Σ consisting of the MVD *Employee* $\twoheadrightarrow$ *SSNo*, and FDs *SSNo* $\to$ *Employee, PassNo*, and *PassNo* $\to$ *Employee, SSNo*. The relation r below satisfies Σ.

relation r

Employee	*SSNo*	*PassNo*
Dag	M02	O38
Digedag	M03	O39

table t_1

Employee	*SSNo*	*PassNo*
Digedag	`ni`	O38
Digedag	`ni`	O39

table t_2

Employee	*SSNo*	*PassNo*
Digedag	M03	O39
Digedag	M03	O39

No data value occurrence in r is *redundant*: if we conceal any single value, then the remaining values and Σ do not uniquely determine the concealed value. □

Commercial database systems deviate from the relational model of data. In the data definition and query standard *SQL* [5] database instances are tables where the column headers of the table correspond to attributes. The rows of the table correspond to tuples, but a table can contain different rows that have the same value in every column. Hence, an SQL table is a *bag* of rows. This feature lowers the cost of data processing as duplicate elimination is considered expensive. Furthermore, a so-called *null value*, marked `ni`, can occur in any column of any row in an SQL table. The null value indicates either non-existing, or existing but unknown, information. This feature of SQL makes it easy to enter new information into the database, since information is not always complete in practice. Null value occurrences can be forbidden for entire columns by declaring the corresponding column header `NOT NULL`. For example, every column of a primary key is `NOT NULL` by default. We now revisit Example 1.

Example 2. Consider the SQL table definition WORK from Example 1 with the same set Σ of constraints and where the column headers *Employee* and *PassNo* are `NOT NULL`. The table t_1 from Example 1 satisfies Σ, according to the definitions of FDs and MVDs in the context of the null value `ni` [2,20]. In particular, t_1 illustrates that the FD *Employee* $\to$ *PassNo* is not implied by Σ in the presence of the `NOT NULL` constraints. Therefore, the `NOT NULL` constraints have a decisive impact on the definition of a suitable normal form for SQL table definitions.

Suppose now we specify all column headers *Employee*, *SSNo* and *PassNo* `NOT NULL`. Then table t_2 also satisfies Σ, and the FD *Employee* $\to$ *SSNo, PassNo* is implied by Σ in the presence of the `NOT NULL` constraints. WORK satisfies the original criteria to be in 4NF [10] with respect to Σ. However, these criteria are still insufficient to guarantee the absence of redundant data value occurrences. In fact, every individual data value occurrence in the table t_2 can be determined by the remaining data values and Σ. Therefore, duplicates have also a decisive impact on the definition of a suitable normal form for SQL table definitions. □

Another important class of constraints over tables are *uniqueness constraints* (UCs). The UC *unique*(X) restricts tables to those that do not have two different rows that are non-null and equal on every column in X. In the relational model UCs are not studied separately because any relation over R satisfies the UC *unique*(X) if and only if it satisfies the FD $X \to R$. However, this equivalence does no longer hold over SQL table definitions T, as illustrated in Example 2. Indeed, if $X = \{Employee\}$, then table t_2 satisfies $X \to T$, but not *unique*(X). This means that, in the context of SQL tables, the combined class of UCs, FDs and MVDs should be studied in the presence of `NOT NULL` constraints. Moreover, Example 2 motivates our pursuit of a normal form condition for SQL table definitions that eliminates redundant data value occurrences.

Contributions and Organization. We summarize previous work in Section 2. In Section 3 we point the reader to related work on constraints by Bernhard Thalheim, to whom this volume is dedicated. We give basic definitions in Section 4. In Section 5 we establish a finite axiomatization for the combined class of UCs, FDs, and MVDs in the presence of `NOT NULL` constraints, and show that the associated decision problem can be decided in almost linear input time. In Section 6 we show that the implication problem of this class is equivalent to that of a fragment in Cadoli and Schaerf's para-consistent family of $\mathcal{S}$-3 logics. In Section 7 we propose a new syntactic normal form condition for SQL table definitions. Finally, in Section 8 we justify our condition semantically by showing that it is necessary and sufficient for the absence of redundant data value occurrences in any SQL tables. We also show that our condition can be checked in time cubic in the input, and is independent of the representation of the constraints. We conclude in Section 10.

2 Related Work

Data dependencies and normalization are essential to the design of the target database, the maintenance of the database during its lifetime, and all major data processing tasks, cf. [36,42].

In the relational model, a UC *unique*(X) over relation schema R is satisfied by a relation if and only if the relation satisfies the FD $X \to R$. Hence, in this context it suffices to study the class of FDs alone. Beeri, Fagin and Howard established the first axiomatization for FDs and MVDs [3]. The associated implication problem can be decided in time almost-linear in the input [11]. Fagin [10] introduced the Fourth normal form for relation schemata. Vincent showed that 4NF is a sufficient and necessary condition to eliminate all possible redundant data value occurrences as well as several data processing difficulties in terms of FDs and MVDs [45]. Arenas and Libkin also justified the 4NF condition in terms of information-theoretic measures [1].

One of the most important extensions of the relational model [4] is incomplete information [18]. This is mainly due to the high demand for the correct handling of such information in real-world applications. While there are several possible interpretations of a null value, many of the previous work on data dependencies is based on Zaniolo's no information interpretation [47]. Atzeni and Morfuni established an axiomatization of FDs in the presence of `NOT NULL` constraints under the no information interpretation [2]. They did not consider bags, which commonly appear in SQL, nor normalization. Köhler and Link investigated UCs and FDs over bags, but considered neither null values nor MVDs [19]. Finally, Hartmann and Link established the equivalence of the implication problem for the combined class of FDs and MVDs in the presence of `NOT NULL` constraints to that of a propositional fragment of Cadoli and Schaerf's family of $\mathcal{S}$-3 logics [16]. However, they only looked at relations where UCs are subsumed by FDs and did not consider neither bags nor normalization. Our equivalences cover those by Sagiv et al. [24] established for the special case where $\mathcal{S}$ covers all variables. SQL-like Armstrong databases have also been studied recently for classes of keys [13] and functional dependencies [12].

3 Some of β's Related Work on Database Constraints

*B*ernhard *Tha*lheim has contributed largely to the theory of data dependencies and database design. His 1991 book on dependencies in relational databases [36] has had over 100 citations by the beginning of 2011. It provides a summary of nearly 100 different classes of data dependencies. The book also contains several of β's own results. These include an axiomatization of full join dependencies [30]. This class subsumes the class of MVDs, and forms an important sub-class of join dependencies which are not finitely Hilbert-style axiomatizable, but do enjoy a Gentzen-style axiomatization. Bernhard has been interested in the class of keys in relational [6,28,29], nested [38] and incomplete databases [35]. For the class of functional dependencies, Bernhard investigated independency questions [7], spreadsheet [9] and graphical reasoning [8], as well as their invariance against errors [17]. Finally, Bernhard has also contributed actively to questions in database design [31,32,40,44] and encouraged research in database theory [33,39]. More generally, Bernhard suggested the approach of consistency enforcement in databases and formal program specification languages [21,22,27] to overcome the limitations of triggers [26]. Being the inaugural recipient of the Peter Chen Award in 2008, β is a pioneer and leading researcher in conceptual modeling. His millennium book on *Entity-Relationship modeling* has had over 400 citations after 10 years. Most notably is the mathematical foundation of conceptual modeling that Bernhard has established with his Higher-Order Entity-Relationship model [34,41]. This includes, in particular, the correct handling of constraints on the conceptual level such as cardinality constraints [37] and multivalued dependencies [43].

4 SQL Table Definitions

We summarize the basic notions. Let $\mathfrak{A} = \{H_1, H_2, \ldots\}$ be a (countably) infinite set of distinct symbols, called (column) headers. An *SQL table definition* is a finite non-empty subset T of $\mathfrak{A}$. Each header H of a table definition T is associated with a countably infinite domain $dom(H)$ which represents the possible values that can occur in the column H denotes. To encompass incomplete information every column may have a null value, denoted by $\texttt{ni} \in dom(H)$. The intention of $\texttt{ni}$ is to mean "no information". This interpretation can therefore model non-existing as well as existing but unknown information [2,47].

For header sets X and Y we may write XY for $X \cup Y$. If $X = \{H_1, \ldots, H_m\}$, then we may write $H_1 \cdots H_m$ for X. In particular, we may write simply H to represent the singleton $\{H\}$. A *row* over T (T-row or simply row, if T is understood) is a function $r : T \to \bigcup_{H \in T} dom(H)$ with $r(H) \in dom(H)$ for all $H \in R$. The null value occurrence $r(H) = \texttt{ni}$ associated with a header H in a row r means that no information is available about the header H for the row r. For $X \subseteq T$ let $r[X]$ denote the restriction of the row r over T to X. An *SQL table* t over T is a finite multi-set of rows over R. In particular, a table t over T may contain two rows r_1 and r_2 such that $r_1 \neq r_2$ and $r_1(H) = r_2(H)$ for all $H \in T$. For a row r over T and a set $X \subseteq T$, r is said to be X-total if for all $H \in X$, $r(H) \neq \texttt{ni}$. Similar, a table t over T is said to be X-total, if every row r of t is X-total. A table t over T is said to be a *total table* if it is T-total.

Following the SQL standard a *uniqueness constraint* (UC) over an SQL table definition T is an expression $\mathit{unique}(X)$ where $X \subseteq T$. An SQL table t over T is said to satisfy the uniqueness constraint $\mathit{unique}(X)$ over T ($\models_t \mathit{unique}(X)$) if and only if for all rows $r_1, r_2 \in t$ the following holds: if $r_1 \neq r_2$ and r_1 and r_2 are both X-total, then there is some $H \in X$ such that $r_1(H) \neq r_2(H)$.

Functional dependencies are important for the relational [4] and other data models [14,15,46]. Following Lien [20], a *functional dependency* (FD) over T is a statement $X \to Y$ where $X, Y \subseteq T$. The FD $X \to Y$ over T is satisfied by a table t over T ($\models_t X \to Y$) if and only if for all $r_1, r_2 \in t$ the following holds: if r_1 and r_2 are X-total and $r_1[X] = r_2[X]$, then $r_1[Y] = r_2[Y]$. We call $X \to Y$ *trivial* whenever $Y \subseteq X$, and non-trivial otherwise. For total tables the FD definition reduces to the standard definition of a functional dependency [36], and so is a sound generalization. It is also consistent with the no-information interpretation [2,20].

Following Lien [20], a *multivalued dependency* (MVD) over T is a statement $X \twoheadrightarrow Y$ where $X, Y \subseteq T$. The MVD $X \twoheadrightarrow Y$ over T is satisfied by a table t over T, denoted by $\models_t X \twoheadrightarrow Y$, if and only if for all $r_1, r_2 \in t$ the following holds: if r_1 and r_2 are X-total and $r_1[X] = r_2[X]$, then there is some $r \in t$ such that $r[XY] = r_1[XY]$ and $r[X(R-Y)] = r_2[X(R-Y)]$. We call $X \twoheadrightarrow Y$ *trivial* whenever $Y \subseteq X$ or $XY = T$, and non-trivial otherwise. For total tables the MVD definition reduces to the standard definition of a multivalued dependency [10,23], and so is a sound generalization. It is also consistent with the no-information interpretation [20].

Following Atzeni and Morfuni [2], a *null-free sub-definition* (NFS) over the table definition T is a an expression T_s where $T_s \subseteq T$. The NFS T_s over T is satisfied by a table t over T ($\models_t T_s$) if and only if t is T_s-total. SQL allows the specification of column headers as `NOT NULL`. Hence, the set of headers declared `NOT NULL` forms an NFS over the underlying SQL table definition.

For a set Σ of constraints over some table definition T, we say that a table t over T *satisfies* Σ ($\models_t \Sigma$) if t satisfies every $\sigma \in \Sigma$. If for some $\sigma \in \Sigma$ the table t does not satisfy σ we say that t violates σ (and violates Σ) and write $\not\models_t \sigma$ ($\not\models_t \Sigma$). We are interested in the combined class $\mathcal{C}$ of UCs, FDs and MVDs in the presence of an NFS.

Constraints interact with one another. Let T be an SQL table definition, let $T_s \subseteq T$ denote an NFS over T, and let $\Sigma \cup \{\varphi\}$ be a set of UCs, FDs and MVDs over T. We say that Σ *implies* φ in the presence of T_s ($\Sigma \models_{T_s} \varphi$) if every table t over T that satisfies Σ and T_s also satisfies φ. If Σ does not imply φ in the presence of T_s we may also write $\Sigma \not\models_{T_s} \varphi$. For Σ we let $\Sigma^*_{T_s} = \{\varphi \mid \Sigma \models_{T_s} \varphi\}$ be the *semantic closure* of Σ, i.e., the set of all UCs, FDs and MVDs implied by Σ in the presence of T_s. In order to determine the logical consequences we use a syntactic approach by applying inference rules, e.g. those in Table 1. These inference rules have the form

$$\frac{\text{premise}}{\text{conclusion}}\,\text{condition},$$

and inference rules without any premise are called axioms. An inference rule is called sound, if whenever the set of constraints in the premise of the rule and the NFS are satisfied by some table over T and the constraints and NFS satisfy the conditions of the rule, then the table also satisfies the constraint in the conclusion of the rule. We let $\Sigma \vdash_{\mathfrak{R}} \varphi$ denote the *inference* of φ from Σ by $\mathfrak{R}$. That is, there is some sequence $\gamma = [\sigma_1, \ldots, \sigma_n]$ of constraints such that $\sigma_n = \varphi$ and every σ_i is an element of Σ or results from an application of an inference rule in $\mathfrak{R}$ to some elements in $\{\sigma_1, \ldots, \sigma_{i-1}\}$. For a finite set Σ, let $\Sigma^+_{\mathfrak{R}} = \{\varphi \mid \Sigma \vdash_{\mathfrak{R}} \varphi\}$ be its *syntactic closure* under inferences by $\mathfrak{R}$. A set $\mathfrak{R}$ of inference rules is said to be *sound* (*complete*) for the implication of UCs, FDs and MVDs in the presence of an NFS if for every table definition T, for every NFS T_s over T and for every set Σ of UCs, FDs and MVDs over T we have $\Sigma^+_{\mathfrak{R}} \subseteq \Sigma^*_{T_s}$ ($\Sigma^*_{T_s} \subseteq \Sigma^+_{\mathfrak{R}}$). The (finite) set $\mathfrak{R}$ is said to be a (finite) *axiomatization* for the implication of UCs, FDs and MVDs in the presence of an NFS if $\mathfrak{R}$ is both sound and complete.

Example 3. The table t_2 in Example 1 satisfies the FD *Employee* $\rightarrow$ *SSNo,PassNo*, but violates the UC *unique*(*Employee*). The table t_1 in Example 1 satisfies the NFS {*Employee,PassNo*}, the UC *unique*(*Employee, SSNo*), the FDs *Employee* $\rightarrow$ *SSNo* and *SSNo* $\rightarrow$ *PassNo*. The table violates the NFS {*SSNo*}, the UC *unique*(*Employee*) and the FD *Employee* $\rightarrow$ *PassNo*. □

5 Axiomatic and Algorithmic Characterization

Let $\mathfrak{B}$ denote the set of inference rules in Table 1. The soundness of the rules in $\mathfrak{B}$ is not difficult to show. For the completeness of $\mathfrak{B}$ we use the result that the set $\mathfrak{D}$ resulting from $\mathfrak{B}$ by removing the FD implication and null pullback rules is sound and complete for FDs and MVDs in the presence of an NFS [16]. In fact, the completeness of $\mathfrak{B}$ follows from that of $\mathfrak{D}$ and the following lemma.

For a set Σ_{UC} of UCs and a set Σ' of FDs and MVDs over table definition T let $\Sigma_{\mathrm{UC}}^{\mathrm{FD}} = \{X \to T \mid \mathit{unique}(X) \in \Sigma_{\mathrm{UC}}\}$ be the set of FDs associated with Σ_{UC} and let $\Sigma[\mathrm{FM}] := \Sigma_{\mathrm{UC}}^{\mathrm{FD}} \cup \Sigma'$ be the set of FDs and MVDs associated with $\Sigma = \Sigma_{\mathrm{UC}} \cup \Sigma'$.

Lemma 1. *Let T be an SQL table definition, T_s an NFS, and Σ a set of UCs, FDs and MVDs over T. Then the following hold:*

1. *$\Sigma \models_{T_s} X \to Y$ if and only if $\Sigma[FM] \models_{T_s} X \to Y$,*
2. *$\Sigma \models_{T_s} X \twoheadrightarrow Y$ if and only if $\Sigma[FM] \models_{T_s} X \twoheadrightarrow Y$,*
3. *$\Sigma \models_{T_s} \mathit{unique}(X)$ if and only if $\Sigma[FM] \models_{T_s} X \to T$ and there is some $\mathit{unique}(Z) \in \Sigma$ such that $Z \subseteq XT_s$.* □

Table 1. Axiomatization $\mathfrak{B}$ of UCs, FDs and MVDs in the presence of an NFS T_s

$$\frac{}{XY \to Y} \text{ (reflexivity, } \mathcal{R}_{\mathrm{F}}) \qquad \frac{X \to YZ}{X \to Y} \text{ (decomposition, } \mathcal{D}_{\mathrm{F}})$$

$$\frac{X \to Y \qquad X \to Z}{X \to YZ} \text{ (FD union, } \mathcal{U}_{\mathrm{F}})$$

$$\frac{X \twoheadrightarrow Y}{X \twoheadrightarrow R - Y} \text{ (}R\text{-complementation, } \mathcal{C}_{\mathrm{M}}^{T}) \qquad \frac{X \twoheadrightarrow Y \qquad X \twoheadrightarrow Z}{X \twoheadrightarrow YZ} \text{ (MVD union, } \mathcal{U}_{\mathrm{M}})$$

$$\frac{X \twoheadrightarrow W \qquad Y \twoheadrightarrow Z}{X \twoheadrightarrow Z - W}\, Y \subseteq X(W \cap T_s) \text{ (null pseudo-transitivity, } \mathcal{T}_{\mathrm{M}})$$

$$\frac{\mathit{unique}(X)}{X \to Y} \text{ (FD implication, } \mathcal{I}_{\mathrm{UF}}) \qquad \frac{X \to Y \quad \mathit{unique}(Y)}{\mathit{unique}(X)}\, Y \subseteq XT_s \text{ (null pullback, } \mathcal{P}_{\mathrm{UF}})$$

$$\frac{X \to Y}{X \twoheadrightarrow Y} \text{ (MVD implication, } \mathcal{I}_{\mathrm{FM}}) \qquad \frac{X \twoheadrightarrow W \qquad Y \to Z}{X \to Z - W}\, Y \subseteq X(W \cap T_s) \text{ (null mixed pseudo-transitivity, } \mathcal{T}_{\mathrm{FM}})$$

Theorem 1. *The set $\mathfrak{B}$ is a finite axiomatization for the implication of UCs, FDs and MVDs in the presence of an NFS.* □

Lemma 1 establishes an algorithmic characterization of the associated implication problem. In fact, it suffices to compute the *header set closure* $X^*_{\Sigma[\mathrm{FM}],T_s} := \{H \in T \mid \Sigma[\mathrm{FM}] \vdash_{\mathfrak{D}} X \to H\}$ and the *dependency basis* $\mathit{DepB}_{\Sigma[\mathrm{FM}],T_s}(X)$

of X with respect to Σ[FM] and T_s [16]. In particular, $DepB_{\Sigma[\mathrm{FM}],T_s}(X)$ is the set of atoms for the Boolean algebra $(Dep(X), \subseteq, \cup, \cap, (\cdot)^{\mathcal{C}}_{T}, \emptyset, T)$ where $Dep(X) = \{Y \subseteq T \mid \Sigma[\mathrm{FM}] \vdash_{\mathfrak{D}} X \twoheadrightarrow Y\}$. The size $||\varphi||$ of φ is the total number of attributes occurring in φ, and the size $||\Sigma||$ of Σ is the sum of $||\sigma||$ over all elements $\sigma \in \Sigma$. For a set Σ of FDs and MVDs let k_Σ denote the number of MVDs in Σ, p_Σ denote the number of sets in the dependency basis $DepB_{\Sigma,T_s}(X)$ of X with respect to Σ and T_s, $\bar{p}_\Sigma$ denote the number of sets in $DepB_{\Sigma,T_s}(X)$ that have non-empty intersection with the right-hand side of φ, and $\Sigma[XT_s]$ denote the set of FDs and MVDs in Σ where the left-hand side is a subset of XT_s. The following result follows from Lemma 1 and the upper time bound established by Galil for relational databases [11].

Theorem 2. *Let φ denote either the UC unique(X), the FD $X \to Y$, or the MVD $X \twoheadrightarrow Y$ over the SQL table definition T. The problem whether φ is implied by a set Σ of UCs, FDs and MVDs in the presence of an NFS T_s over T can be decided in $\mathcal{O}(||\Sigma|| + \min\{k_{\Sigma[FM][XT_s]}, \log \bar{p}_{\Sigma[FM][XT_s]}\} \times ||\Sigma[FM][XT_s]||)$ time.* □

Example 4. Let T denote the SQL table definition with column headers *Employee*, *SSNo* and *PassNo*, constraint set Σ with the MVD *Employee* $\twoheadrightarrow$ *SSNo*, the FD *PassNo* $\to$ *SSNo* and the UC *unique*(SSNo). Furthermore, we have the NFS $T_s = \{SSNo, PassNo\}$. The NFS allows us to apply the null pullback rule $\mathcal{P}_{\mathrm{UF}}$ to the given FD and UC to infer the UC *unique*(PassNo). We can apply the FD implication rule $\mathcal{I}_{\mathrm{UF}}$ to the given UC to infer the FD *SSNo* $\to$ *PassNo*. The NFS allows us to apply the null mixed pseudo-transitivity rule $\mathcal{T}_{\mathrm{FM}}$ to the given MVD and the FD *SSNo* $\to$ *PassNo* to infer the FD *Employee* $\to$ *PassNo*. The NFS allows us to apply the null pullback rule $\mathcal{P}_{\mathrm{UF}}$ to the FD *Employee* $\to$ *PassNo* and UC *unique*(PassNo) to infer the UC *unique*(Employee). By the soundness of the inference rules in $\mathfrak{B}$ it follows that both UCs *unique*(PassNo) and *unique*(Employee) are implied by Σ in the presence of T_s. □

6 Logical Characterization: Equivalence to $\mathcal{S}$-3 Implication

Here we refine the correspondence between the implication of FDs and MVDs in the presence of NFSs and the implication of a fragment in Cadoli and Schaerf's family of $\mathcal{S}$-3 logics, established for tables that are sets of rows [16].

$\mathcal{S}$-3 semantics. Schaerf and Cadoli [25] introduced $\mathcal{S}$-3 logics as "a semantically well-founded logical framework for sound approximate reasoning, which is justifiable from the intuitive point of view, and to provide fast algorithms for dealing with it even when using expressive languages".

For a finite set $\mathcal{L}$ of propositional variables let $\mathcal{L}^\ell$ denote the set of all literals over $\mathcal{L}$, i.e., $\mathcal{L}^\ell = \mathcal{L} \cup \{\neg H' \mid H' \in \mathcal{L}\} \subseteq \mathcal{L}^*$ where $\mathcal{L}^*$ denotes the propositional language over $\mathcal{L}$. Let $\mathcal{S} \subseteq \mathcal{L}$. An $\mathcal{S}$-3 interpretation of $\mathcal{L}$ is a total function $\hat{\omega} : \mathcal{L}^\ell \to \{\mathbb{F}, \mathbb{T}\}$ that maps every variable $H' \in \mathcal{S}$ and its negation $\neg H'$ into opposite values ($\hat{\omega}(H') = \mathbb{T}$ if and only if $\hat{\omega}(\neg H') = \mathbb{F}$), and that does not map both a variable $H' \in \mathcal{L} - \mathcal{S}$ and its negation $\neg H'$ into $\mathbb{F}$ (we must not have $\hat{\omega}(H') = \mathbb{F} = \hat{\omega}(\neg H')$ for any $H' \in \mathcal{L} - \mathcal{S}$). An $\mathcal{S}$-3 interpretation $\hat{\omega} : \mathcal{L}^{\ell} \to \{\mathbb{F}, \mathbb{T}\}$ of $\mathcal{L}$ can be lifted to a total function $\hat{\Omega} : \mathcal{L}^* \to \{\mathbb{F}, \mathbb{T}\}$ by means of simple rules [25]. Since we are only interested in special formulae that are all in Negation Normal Form we require only the following three rules for assigning truth values: (1) $\hat{\Omega}(\varphi') = \hat{\omega}(\varphi')$, if $\varphi' \in \mathcal{L}^{\ell}$, (2) $\hat{\Omega}(\varphi' \vee \psi') = \mathbb{T}$, if $\hat{\Omega}(\varphi') = \mathbb{T}$ or $\hat{\Omega}(\psi') = \mathbb{T}$, and (3) $\hat{\Omega}(\varphi' \wedge \psi') = \mathbb{T}$, if $\hat{\Omega}(\varphi') = \mathbb{T}$ and $\hat{\Omega}(\psi') = \mathbb{T}$. An $\mathcal{S}$-3 interpretation $\hat{\omega}$ is a *model* of a set Σ' of $\mathcal{L}$-formulae, if $\hat{\Omega}(\sigma') = \mathbb{T}$ holds for every $\sigma' \in \Sigma'$. We say that Σ' *$\mathcal{S}$-3 implies* an $\mathcal{L}$-formula φ', denoted by $\Sigma' \models_{\mathcal{S}}^{3} \varphi'$, if every $\mathcal{S}$-3 interpretation that is a model of Σ' is also a model of φ'.

Mappings between constraints and formulae. In the first step, we define the fragment of $\mathcal{L}$-formulae that corresponds to UCs, FDs, MVDs in the presence of an NFS T_s over a table definition T. Let $\phi : T \to \mathcal{L}$ denote a bijection between T and the set $\mathcal{L} = \{H' \mid H \in T\}$ of propositional variables that corresponds to T. For an NFS T_s over T let $\mathcal{S} = \phi(T_s)$ be the set of propositional variables in $\mathcal{L}$ that corresponds to T_s. Hence, the variables in $\mathcal{S}$ are the images of those column headers of T declared `NOT NULL`. We now extend ϕ to a mapping Φ from the set of UCs, FDs and MVDs over T. For a UC $unique(H_1, \ldots, H_n)$ over T, let $\Phi(unique(H_1, \ldots, H_n))$ denote the goal clause $\neg H_1' \vee \cdots \vee \neg H_n'$. For an FD $H_1, \ldots, H_n \to H$ over T, let $\Phi(H_1, \ldots, H_n \to H)$ denote the definite clause $\neg H_1' \vee \cdots \vee \neg H_n' \vee H'$. Finally, for an MVD $H_1, \ldots, H_n \twoheadrightarrow F_1, \ldots, F_m$ over T, let $\Phi(H_1, \ldots, H_n \twoheadrightarrow F_1, \ldots, F_m)$ denote the formula $\neg H_1' \vee \cdots \vee \neg H_n' \vee (F_1' \wedge \cdots \wedge F_m') \vee (G_1' \wedge \cdots \wedge G_k')$ where $T = \{H_1, \ldots, H_n, F_1, \ldots, F_m, G_1, \ldots, G_k\}$. For the sake of presentation, but without loss of generality, we assume that FDs have only a single column header on their right-hand side. As usual, disjunctions (conjunctions) over zero disjuncts (conjuncts) are interpreted as $\mathbb{F}$ ($\mathbb{T}$). In what follows, we may simply denote $\Phi(\varphi) = \varphi'$ and $\Phi(\Sigma) = \{\sigma' \mid \sigma \in \Sigma\} = \Sigma'$.

The equivalence. Our aim is to show that for every SQL table definition T, for every set $\Sigma \cup \{\varphi\}$ of UCs, FDs and MVDs and for every NFS T_s over T, there is some T_s-total table t that satisfies Σ and violates φ if and only if there is an $\mathcal{S}$-3 model $\hat{\omega}_t$ of Σ' that is not an $\mathcal{S}$-3 model of φ'. For an arbitrary table t it is not obvious how to define the $\mathcal{S}$-3 interpretation $\hat{\omega}_t$. However, for deciding the implication problem $\Sigma \models_{T_s} \varphi$ it suffices to examine two-row tables, instead of arbitrary tables. For two-row tables $\{r_1, r_2\}$ we define the *special-3-interpretation* of $\mathcal{L}$ by

- $\hat{\omega}_{\{r_1,r_2\}}(H') = \mathbb{T}$ and $\hat{\omega}_{\{r_1,r_2\}}(\neg H') = \mathbb{F}$, if $\mathtt{ni} \neq r_1(H) = r_2(H) \neq \mathtt{ni}$,
- $\hat{\omega}_{\{r_1,r_2\}}(H') = \mathbb{T}$ and $\hat{\omega}_{\{r_1,r_2\}}(\neg H') = \mathbb{T}$, if $r_1(H) = \mathtt{ni} = r_2(H)$,
- $\hat{\omega}_{\{r_1,r_2\}}(H') = \mathbb{F}$ and $\hat{\omega}_{\{r_1,r_2\}}(\neg H') = \mathbb{T}$, if $r_1(H) \neq r_2(H)$

for all $H' \in \mathcal{L}$. If $\{r_1, r_2\}$ is T_s-total, then $\hat{\omega}_{\{r_1,r_2\}}$ is an $\mathcal{S}$-3 interpretation.

Theorem 3. *Let $\Sigma \cup \{\varphi\}$ be a set of UCs, FDs and MVDs over the SQL table definition T, and let T_s denote an NFS over T. Let $\mathcal{L}$ denote the set of propositional variables that corresponds to T, $\mathcal{S}$ the set of variables that corresponds to T_s, and $\Sigma' \cup \{\varphi'\}$ the set of formulae over $\mathcal{L}$ that correspond to $\Sigma \cup \{\varphi\}$. Then $\Sigma \models_{T_s} \varphi$ if and only if $\Sigma' \models_{\mathcal{S}}^{3} \varphi'$.* □

Example 5. Consider the table definition T, NFS T_s and constraint set Σ from Example 4. Suppose we wonder if the UCs $\varphi_1 = \mathit{unique}(\text{PassNo})$ and $\varphi_2 = \mathit{unique}(\text{Employee})$ are implied by Σ in the presence of T_s. According to Theorem 3 the problems $\Sigma \models_{T_s} \varphi_1$ and $\Sigma \models_{T_s} \varphi_2$ are equivalent to $\Sigma' \models^3_{\mathcal{S}} \varphi'_1$ and $\Sigma' \models^3_{\mathcal{S}} \varphi'_2$ where $\mathcal{S} = \{\mathit{SSNo}', \mathit{PassNo}'\}$.

Suppose an $\mathcal{S}$-3 interpretation $\hat{\omega}$ is not a model of φ'_1. Then $\hat{\omega}(\neg \mathit{PassNo}') = \mathbb{F}$. For $\hat{\omega}$ to be an $\mathcal{S}$-3 model of Σ' we must thus have $\hat{\omega}(\mathit{SSNo}') = \mathbb{T} = \hat{\omega}(\neg \mathit{SSNo}')$, but $\mathit{SSNo}' \in \mathcal{S}$. We conclude that $\Sigma' \models^3_{\mathcal{S}} \varphi'_1$ and by Theorem 3 also $\Sigma \models_{T_s} \varphi_1$.

Suppose an $\mathcal{S}$-3 interpretation $\hat{\omega}$ is not a model of φ'_2. Then $\hat{\omega}(\neg \mathit{Employee}') = \mathbb{F}$. For $\hat{\omega}$ to be an $\mathcal{S}$-3 model of Σ' we must thus have $\hat{\omega}(\mathit{SSNo}') = \mathbb{T}$ or $\hat{\omega}(\neg \mathit{PassNo}') = \mathbb{T}$. Moreover, for $\hat{\omega}$ to be an $\mathcal{S}$-3 model of Σ' we must also have $\hat{\omega}(\neg \mathit{SSNo}') = \mathbb{T} = \hat{\omega}(\mathit{PassNo}')$. However, $\mathit{SSNo}', \mathit{PassNo}' \in \mathcal{S}$. We conclude that $\Sigma' \models^3_{\mathcal{S}} \varphi'_2$ and by Theorem 3 also $\Sigma \models_{T_s} \varphi_2$. □

Example 6. Let T denote the SQL table definition with column headers *Employee*, *SSNo* and *PassNo*, constraint set Σ with the MVD $\mathit{Employee} \twoheadrightarrow \mathit{SSNo}$, and the UCs *unique*(SSNo) and *unique*(PassNo). Furthermore, we have the NFS $T_s = \{\mathit{SSNo}\}$. For $\varphi = \mathit{unique}(\mathit{Employee})$, $\Sigma \not\models_{T_s} \varphi$ as the following SQL table t demonstrates:

Employee	*SSNo*	*PassNo*
Digedag	M03	`ni`
Digedag	M01	`ni`

.

Indeed, the special $\mathcal{S}$-3 interpretation $\hat{\omega}_t$ where for all $L \in \mathcal{L}^\ell$, $\hat{\omega}_t(L) = \mathbb{F}$ iff $L \in \{\neg \mathit{Employee}', \mathit{SSNo}'\}$ is a model of Σ' but not a model of φ'. □

7 The Fourth Normal Form for SQL Table Definitions

Fagin [10] introduced a normal form condition on relation schemata that characterizes the absence of redundant data value occurrences in any relation over the schema, caused by FDs and MVDs [45]. To the best of our knowledge no such normal form has been proposed yet for SQL table definitions.

Definition 1. *Let T denote an SQL table definition, T_s a null-free subdefinition, and Σ a set of UCs, FDs and MVDs over T. Then T is said to be in* Fourth Normal Form *(4NF) with respect to Σ and T_s if and only if for all non-trivial multivalued dependencies $X \twoheadrightarrow Y \in \Sigma^+_{\mathfrak{B}}$ we have $\mathit{unique}(X) \in \Sigma^+_{\mathfrak{B}}$.* □

Note that our axiomatization $\mathfrak{B}$ enables us to state the 4NF condition in purely syntactic terms. As we will see later, this results in a syntactic characterization for the absence of redundant data value occurrences. Moreover, since the 4NF condition is defined with respect to the syntactic closure, the property of being in 4NF is independent of the representation of the given set of UCs, FDs and MVDs. That is, for every set Σ' that is equivalent to Σ it is true that T is in 4NF with respect to Σ and T_s if and only if T is in 4NF with respect to Σ' and T_s. Definition 1 subsumes Fagin's classic 4NF condition for relation schemata, which is the special case where $T_s = T$ and no duplicate rows are allowed to occur in a table over T. Finally, every SQL table definition that is in 4NF with respect to a set Σ of UCs, FDs, and MVDs and an NFS T_s over T is also in Boyce-Codd-Heath Normal Form with respect to Σ and T_s. That is, for every non-trivial FD $X \to Y \in \Sigma^+_{\mathfrak{B}}$ we have $X \twoheadrightarrow Y \in \Sigma^+_{\mathfrak{B}}$ by an application of the implication rule $\mathcal{I}_{\mathrm{FM}}$, and thus $unique(X) \in \Sigma^+_{\mathfrak{B}}$ by the 4NF criteria.

Example 7. The SQL table definition of Example 5 is indeed in 4NF with respect to the given constraint set Σ and the given NFS T_s. However, the SQL table definition of Example 6 is not in 4NF with respect to the given constraint set Σ and the given NFS T_s. □

8 Semantic Justification of 4NF

We will now justify our syntactic definition of 4NF semantically by showing that the condition is sufficient and necessary for the absence of redundant data value occurrences in any future tables. Following Vincent [45] we will make the notion of data redundancy explicit. Let T be an SQL table definition, H a column header of T, and r a row over T. A *replacement* of $r(H)$ is a row $\bar{r}$ over T that satisfies the following conditions: i) for all $\bar{H} \in T - \{H\}$ we have $\bar{r}(\bar{H}) = r(\bar{H})$, and ii) $\bar{r}(H) \neq r(H)$. Intuitively, a data value occurrence in some Σ-satisfying table is redundant if the occurrence cannot be replaced by any other data value without violating some constraint in Σ.

Definition 2. *Let T be an SQL table definition, $H \in T$ a column header, T_s an NFS and Σ a set of UCs, FDs and MVDs over T, t a table over T that satisfies Σ and T_s, and r a row in t. We say that the data value occurrence $r(H)$ is* redundant *if and only if* every *replacement $\bar{r}$ of $r(H)$ results in a table $\bar{t} := (t - \{r\}) \cup \{\bar{r}\}$ that violates Σ. We say that T is in* Redundancy-Free Normal Form *(RFNF) with respect to Σ and T_s if and only if there is no table t over T such that i) t satisfies Σ and T_s, and ii) t contains a row r such that for some column header H of T the data value occurrence $r(H)$ is redundant.* □

We show that the syntactic 4NF condition of Definition 1 captures the semantic RFNF condition of Definition 2.

Theorem 4. *Let T be an SQL table definition, T_s an NFS and Σ a set of UCs, FDs and MVDs over T. Then T is in RFNF with respect to Σ and T_s if and only if T is in 4NF with respect to Σ and T_s.* □

Example 8. Consider again Example 6. Here, the SQL table definition is not in 4NF with respect to Σ and T_s. In particular, *unique*(*Employee*) is not implied by Σ in the presence of T_s. According to Theorem 4, T is not in RFNF with respect to Σ and T_s. Indeed, both value occurrences in the column *PassNo* of the table t in Example 6 are redundant. □

Definition 1 refers to the syntactic closure $\Sigma^+_{\mathfrak{B}}$ of Σ and T_s under $\mathfrak{B}$, which can be exponential in the size of Σ. Therefore, the question remains if the problem whether an SQL table definition is in 4NF with respect to Σ and T_s can be decided efficiently.

Theorem 5. *Let T be an SQL table definition, T_s an NFS and Σ a set of UCs, FDs and MVDs over T. Then the following conditions are equivalent:*

1. *T is in 4NF with respect to Σ and T_s,*
2. *for all non-trivial FDs $X \to Y \in \Sigma$ and for all non-trivial MVDs $X \twoheadrightarrow Y \in \Sigma$ we have: unique$(X) \in \Sigma^+_{\mathfrak{B}}$,*
3. *for all non-trivial FDs $X \to Y \in \Sigma$ and for all non-trivial MVDs $X \twoheadrightarrow Y \in \Sigma$ we have: $X \to T \in \Sigma^+_{\mathfrak{B}}$ and there is some unique$(Z) \in \Sigma$ such that $Z \subseteq XT_s$.* □

The following result follows directly from Theorem 5 and Theorem 2.

Theorem 6. *The problem whether an SQL table definition T is in Fourth Normal Form with respect to a set Σ of UCs, FDs and MVDs, and an NFS T_s over T can be decided in $\mathcal{O}(||\Sigma||^2 \times |\Sigma|)$ time.* □

9 Challenges with Database Normalization

Finally, we will comment on the impact of duplicate and partial information on achieving the 4NF condition. We start with some comments about well-known facts from relational databases. If a relation satisfies the FD $X \to Y$, then the relation is the *lossless join* of its projection on the attribute sets XY and $X(R-Y)$. Hence, FDs provide a condition that is sufficient for lossless *decompositions* of relations. MVDs provide a sufficient and necessary condition for relations to be decomposable into two of its projections, i.e., a relation satisfies the MVD $X \twoheadrightarrow Y$ if and only if the relation is the lossless join of its projections on XY and $X(R-Y)$. Indeed, this property was the original motivation for introducing the concept of multivalued dependencies [10].

For tables we say that a row $\bar{r}$ is *subsumed* by the row r, if for every column header H, $\bar{r}(H) = \texttt{ni}$ or $\bar{r}(H) = r(H)$ [20]. If tables are restricted to those that are subsumption-free, i.e. do not contain any two rows where one subsumes the other, then the lossless decomposition properties are still valid for the X-total sub-tables. That is, the X-total sub-table of a subsumption-free table that satisfies the FD $X \to Y$ is the lossless join of the X-total sub-tables of its projections on XY and $X(R-Y)$. Furthermore, the X-total sub-table of a subsumption-free table satisfies the MVD $X \twoheadrightarrow Y$ if and only if its X-total sub-table is the lossless join of the X-total sub-tables of its projections on XY and $X(R-Y)$. Since projections of subsumption-free tables may not be subsumption-free, subsumed rows are removed in this case [20]. These properties provide a nice generalization of the concepts of FDs and MVDs to partial information that is faithful to their original motivation. SQL table definitions T for which

Σ implies a *key* in the presence of the NFS T_s, i.e. where for some $X \subseteq T$ we have $\Sigma \models_{T_s}$ *unique*(X) and $X \subseteq T_s$, only permit subsumption-free tables. However, SQL does not require the existence of a key and tables can contain rows that subsume one another. In particular, the removal of subsumed (in particular duplicate) rows is considered an expensive operation. This situation raises new challenges for the generalization of classic database normalization techniques. In fact, MVDs do not even provide a sufficient condition for achieving a lossless decomposition, even if different notions of projections are considered.

Example 9. The table t_1 from Example 1 satisfies *Employee* $\twoheadrightarrow$ *SSNo*. If we do not remove subsumed (including duplicate) rows, then we obtain:

projection of t_1 on {*Employee, SSNo*}

Employee	*SSNo*
Digedag	`ni`
Digedag	`ni`

projection of t_1 on {*Employee, PassNo*}

Employee	*PassNo*
Digedag	O38
Digedag	O39

.

A join of these two table projections results in a table different from t_1. Consider the table t_2 below that satisfies the MVD *Employee* $\twoheadrightarrow$ *SSNo*. If we do remove subsumed rows, then we obtain:

table t_2

Employee	*SSNo*	*PassNo*
Digedag	M02	O38
Digedag	`ni`	O38
Digedag	`ni`	O39
Digedag	M02	O39

projection of t_2 on {*Employee, SSNo*}

Employee	*SSNo*
Digedag	MO2

projection of t_2 on {*Employee, PassNo*}

Employee	*PassNo*
Digedag	O38
Digedag	O39

.

A join of these two table projections results in a table different from t_2. Thus, subsumed rows must be removed from projections only sometimes. □

The example illustrates that the decomposition approach to database normalization requires new attention when we consider the features of SQL. The presence of duplicates requires UCs in addition to FDs, but UCs are not preserved when performing joins. Hence, it is not clear what *dependency-preservation* means. The presence of null values requires join attributes to be `NOT NULL` when *lossless* decompositions are to be achieved. Furthermore, projection becomes more difficult to define when duplicates (more generally, subsumed rows) are to be eliminated only sometimes.

10 Conclusion

The class of UCs is not subsumed by the class of FDs and MVDs over SQL tables, in contrast to relations. For this purpose, we have characterized the implication problem for the combined class of UCs, FDs and MVDs in the presence of `NOT NULL` constraints axiomatically, algorithmically and logically. We have further proposed a syntactic Fourth Normal Form condition for SQL table definitions, and justified this condition semantically. On one hand, the semantics of SQL really calls for a comprehensive support to specify and maintain FDs and MVDs to guarantee consistency and locate data redundancy. On the other hand, the SQL features motivate a thorough study of database normalization.

References

1. Arenas, M., Libkin, L.: An information-theoretic approach to normal forms for relational and XML data. J. ACM 52(2), 246–283 (2005)
2. Atzeni, P., Morfuni, N.: Functional dependencies and constraints on null values in database relations. Information and Control 70(1), 1–31 (1986)
3. Beeri, C., Fagin, R., Howard, J.H.: A complete axiomatization for fds and mvds in database relations. In: SIGMOD, pp. 47–61. ACM (1977)
4. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM 13(6), 377–387 (1970)
5. Date, C., Darwen, H.: A guide to the SQL standard. Addison-Wesley Professional, Reading (1997)
6. Demetrovics, J., Katona, G., Miklós, D., Seleznjev, O., Thalheim, B.: Asymptotic properties of keys and functional dependencies in random databases. Theor. Comput. Sci. 190(2), 151–166 (1998)
7. Demetrovics, J., Katona, G.O.H., Miklós, D., Thalheim, B.: On the Number of Independent Functional Dependencies. In: Dix, J., Hegner, S.J. (eds.) FoIKS 2006. LNCS, vol. 3861, pp. 83–91. Springer, Heidelberg (2006)
8. Demetrovics, J., Molnár, A., Thalheim, B.: Graphical Reasoning for Sets of Functional Dependencies. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 166–179. Springer, Heidelberg (2004)
9. Demetrovics, J., Molnár, A., Thalheim, B.: Relationship Design Using Spreadsheet Reasoning for Sets of Functional Dependencies. In: Manolopoulos, Y., Pokorný, J., Sellis, T.K. (eds.) ADBIS 2006. LNCS, vol. 4152, pp. 108–123. Springer, Heidelberg (2006)
10. Fagin, R.: Multivalued dependencies and a new normal form for relational databases. ACM Trans. Database Syst. 2(3), 262–278 (1977)
11. Galil, Z.: An almost linear-time algorithm for computing a dependency basis in a relational database. J. ACM 29(1), 96–102 (1982)
12. Hartmann, S., Kirchberg, M., Link, S.: Design by example for SQL table definitions with functional dependencies. The VLDB Journal (2011), doi:10.1007/s00778-011-0239-5
13. Hartmann, S., Leck, U., Link, S.: On Codd families of keys over incomplete relations. Comput. J. 54(7), 1166–1180 (2011)
14. Hartmann, S., Link, S.: Efficient reasoning about a robust XML key fragment. ACM Trans. Database Syst. 34(2) (2009)
15. Hartmann, S., Link, S.: Numerical constraints on XML data. Inf. Comput. 208(5), 521–544 (2010)
16. Hartmann, S., Link, S.: When data dependencies over SQL tables meet the Logics of Paradox and $\mathcal{S}$-3. In: PODS, pp. 317–326 (2010)
17. Hartmann, S., Link, S., Schewe, K.-D.: Weak Functional Dependencies in Higher-Order Datamodels. In: Seipel, D., Turull-Torres, J.M. (eds.) FoIKS 2004. LNCS, vol. 2942, pp. 116–133. Springer, Heidelberg (2004)

18. Imielinski, T., Lipski Jr., W.: Incomplete information in relational databases. J. ACM 31(4), 761–791 (1984)
19. Köhler, H., Link, S.: Armstrong axioms and Boyce-Codd-Heath normal form under bag semantics. Inf. Process. Lett. 110(16), 717–724 (2010)
20. Lien, E.: On the equivalence of database models. J. ACM 29(2), 333–362 (1982)
21. Link, S.: Consistency Enforcement in Databases. In: Bertossi, L., Katona, G.O.H., Schewe, K.-D., Thalheim, B. (eds.) Semantics in Databases 2001. LNCS, vol. 2582, pp. 139–159. Springer, Heidelberg (2003)
22. Link, S., Schewe, K.-D.: An arithmetic theory of consistency enforcement. Acta Cybern. 15(3), 379–416 (2002)
23. Paredaens, J., De Bra, P., Gyssens, M., Van Gucht, D.: The Structure of the Relational Database Model. Springer, Heidelberg (1989)
24. Sagiv, Y., Delobel, C., Parker Jr., D.S., Fagin, R.: An equivalence between relational database dependencies and a fragment of propositional logic. J. ACM 28(3), 435–453 (1981)
25. Schaerf, M., Cadoli, M.: Tractable reasoning via approximation. Artif. Intell. 74, 249–310 (1995)
26. Schewe, K.-D., Thalheim, B.: Limitations of rule triggering systems for integrity maintenance in the context of transition specifications. Acta Cybern. 13(3), 277–304 (1998)
27. Schewe, K.-D., Thalheim, B.: Towards a theory of consistency enforcement. Acta Inf. 36(2), 97–141 (1999)
28. Selesnjev, O., Thalheim, B.: On the numbers of shortes keys in relational databases on non-uniform domains. Acta Cybern. 8, 267–271 (1988)
29. Seleznjev, O., Thalheim, B.: Behavior of keys in random databases. In: SCCC, pp. 171–183 (1998)
30. Thalheim, B.: A compelte axiomatization for full join dependencies in relations. Bulletin of the EATCS 24, 109–114 (1984)
31. Thalheim, B.: Deductive normal forms of relations. In: Mathematical Methods of Specification and Synthesis of Software Systems, pp. 226–230 (1985)
32. Thalheim, B.: Design Tools for Large Relational Database Systems. In: Biskup, J., Demetrovics, J., Paredaens, J., Thalheim, B. (eds.) MFDBS 1987. LNCS, vol. 305, pp. 210–224. Springer, Heidelberg (1988)
33. Thalheim, B.: Open Problems in Database Theory. In: Biskup, J., Demetrovics, J., Paredaens, J., Thalheim, B. (eds.) MFDBS 1987. LNCS, vol. 305, pp. 241–247. Springer, Heidelberg (1988)
34. Thalheim, B.: The Higher-Order Entity-Relationship model and (DB)2. In: Demetrovics, J., Thalheim, B. (eds.) MFDBS 1989. LNCS, vol. 364, pp. 382–397. Springer, Heidelberg (1989)
35. Thalheim, B.: On semantic issues connected with keys in relational databases permitting null values. Elektronische Informationsverarbeitung und Kybernetik 25(1-2), 11–20 (1989)
36. Thalheim, B.: Dependencies in relational databases. Teubner (1991)
37. Thalheim, B.: Fundamentals of Cardinality Constraints. In: Pernul, G., Tjoa, A.M. (eds.) ER 1992. LNCS, vol. 645, pp. 7–23. Springer, Heidelberg (1992)
38. Thalheim, B.: The number of keys in relational and nested relational databases. Discrete Applied Mathematics 40(2) (1992)
39. Thalheim, B.: An overview on database theory. Datenbank Rundbrief 10, 2–13 (1992)
40. Thalheim, B.: Database design strategies. In: CISM, pp. 267–285 (1993)

41. Thalheim, B.: Foundations of Entity - Relationship Modeling. Ann. Math. Artif. Intell. 7(1-4), 197–256 (1993)
42. Thalheim, B.: Entity-Relationship modeling. Springer, Heidelberg (2000)
43. Thalheim, B.: Conceptual Treatment of Multivalued Dependencies. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 363–375. Springer, Heidelberg (2003)
44. Thalheim, B.: Component development and construction for database design. Data Knowl. Eng. 54(1), 77–95 (2005)
45. Vincent, M.: Semantic foundation of 4NF in relational database design. Acta Inf. 36, 1–41 (1999)
46. Vincent, M., Liu, J., Liu, C.: Strong FDs and their application to normal forms in XML. ACM Trans. Database Syst. 29(3), 445–462 (2004)
47. Zaniolo, C.: Database relations with null values. J. Comput. Syst. Sci. 28(1), 142–166 (1984)

Independent Update Reflections on Interdependent Database Views

Stephen J. Hegner

Umeå University, Department of Computing Science
SE-901 87 Umeå, Sweden
hegner@cs.umu.se
http://www.cs.umu.se/~hegner

Abstract. The problem of identifying suitable view-update strategies is typically addressed in the context of a single view. However, it is often the case that several views must co-exist; the challenge is then to find strategies which allow one view to be updated without affecting the other. The classical constant-complement strategy can provide a solution to this problem; however, both the context and the admissible updates are quite limited. In this work, the updates which are possible within this classical approach are extended substantially via a technique which considers only the states which are reachable from a given initial configuration. The results furthermore do not depend upon complementation, and thus are readily extensible to settings involving more than two views.

1 Introduction

Both views and updates are fundamental to a comprehensive database system. Consequently, the problem of how to support updates to views has been studied extensively. Most work addresses this problem in the context of a single view, including the classical approach via the relational algebra [10,18,19,7,8], the more recent approach based upon *database repairs* [1,3,2], and work which bridges these two approaches [12]. However, in some situations a number of distinct yet interdependent views of the same main schema must co-exist. Often, the access rights to these views differ, so that a user or access rôle [4,21] which has access to one view may not even be allowed to read, much less update, another. In such a setting, it is important to identify those updates which are possible to a given view Γ without requiring any access to the other views, for reading or for writing. This may be recaptured succinctly in terms of two independence conditions. First of all, whether or not an update to Γ is to be allowed at all should be independent of the states of the other views. This is called *context independence*. Second, the reflection to the main schema of the update to the selected view must not require a change of the state of any of the other views. This is called *propagation independence* or *locality of effect*. In the presence of these two forms of independence, an update may be made to the given view Γ without knowledge about the states of the other views beyond that which is already known in Γ, and the result of the update to Γ will not be visible in any of the other views. Applications in which such independence is central, and which have motivated this work, include component-based architectures [24,23,13,16], update by cooperation [17], and models of data objects for transactions [15].

For the case of two views, the classical constant-complement approach [6,11] already provides a very elegant solution in the situations to which it applies. Unfortunately, it imposes conditions which are often too strong to be of use, as illustrated by the following examples.

Let $\mathbf{E}_0$ be the relational schema consisting of the single relation symbol $R[ABC]$, constrained by the functional dependency (FD) $B \to C$. Define $\Pi_{AB}^{\mathbf{E}_0} = (\mathbf{E}_0^{AB}, \pi_{AB}^{\mathbf{E}_0})$ to be the view whose schema $\mathbf{E}_0^{AB}$ contains the single relation symbol $R_{AB}[AB]$ and whose morphism $\pi_{AB}^{\mathbf{E}_0}$ is the projection of $R[ABC]$ onto $R_{AB}[AB]$. Define $\Pi_{BC}^{\mathbf{E}_0} = (\mathbf{E}_0^{BC}, \pi_{BC}^{\mathbf{E}_0})$ analogously. Let $\mathsf{LDB}(\mathbf{E}_0)$ denote the set of all *legal databases* of $\mathbf{E}_0$; that is, the set of all relations on $R[ABC]$ which satisfy the FD $B \to C$. Define $\mathsf{LDB}(\mathbf{E}_0^{AB})$ and $\mathsf{LDB}(\mathbf{E}_0^{BC})$ similarly, as the legal databases of the corresponding view schemata. Define the *decomposition mapping* $\pi_{AB}^{\mathbf{E}_0} \times \pi_{BC}^{\mathbf{E}_0} : \mathsf{LDB}(\mathbf{E}_0) \to \mathsf{LDB}(\mathbf{E}_0^{AB}) \times \mathsf{LDB}(\mathbf{E}_0^{BC})$ on elements by $M \mapsto (\pi_{AB}^{\mathbf{E}_0}(M), \pi_{BC}^{\mathbf{E}_0}(M))$.

Let $u_1 = (N_1, N_1')$ be any update on $\Pi_{AB}^{\mathbf{E}_0}$, with N_1 representing the view state before the update operation and N_2 the state afterwards. A *reflection* of u_1 to $\mathbf{E}_0$ is any $(M_1, M_2) \in \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D})$ with $\pi_{AB}^{\mathbf{E}_0}(M_1) = N_1$ and $\pi_{AB}^{\mathbf{E}_0}(M_2) = N_1'$. This update is *propagation independent* with respect to $\Pi_{BC}^{\mathbf{E}_0}$, or *keeps* $\Pi_{BC}^{\mathbf{E}_0}$ *constant*, if $\pi_{BC}^{\mathbf{E}_0}(M_1) = \pi_{BC}^{\mathbf{E}_0}(M_2)$.

In this example, the set of all updates on $\Pi_{AB}^{\mathbf{E}_0}$ which keep the state of $\Pi_{BC}^{\mathbf{E}_0}$ constant has a very simple characterization; namely, it is precisely the set of all updates on R_{AB} which keep the projection onto B fixed. Similarly, the set of all updates on $\Pi_{BC}^{\mathbf{E}_0}$ with $\Pi_{AB}^{\mathbf{E}_0}$ constant is precisely the set of all updates on R_{BC} which keep the projection onto B fixed. The view $\Pi_{B}^{\mathbf{E}_0}$ of $\mathbf{E}_0$ which is the projection onto B, is called the *meet* of $\Pi_{AB}^{\mathbf{E}_0}$ and $\Pi_{BC}^{\mathbf{E}_0}$. For both $\Pi_{AB}^{\mathbf{E}_0}$ and $\Pi_{BC}^{\mathbf{E}_0}$, the updates which are propagation independent are precisely those which keep the meet view $\Pi_{B}^{\mathbf{E}_0}$ constant. Thus, whether or not an update to either view is possible without modifying the state of the other is a property of the state of that view alone, and does not require further knowledge of the state of $\mathbf{E}_0$; i.e., it exhibits context independence. For a more thorough presentation of these ideas in the context of update via constant complement, see [11, 1.2].

Pairs of views are not always so well behaved. Let $\mathbf{E}_1$ be identical to $\mathbf{E}_0$, save that it is governed by the additional FD $A \to C$, and let $\Pi_{AB}^{\mathbf{E}_1}$ and $\Pi_{BC}^{\mathbf{E}_1}$ be defined analogously to $\Pi_{AB}^{\mathbf{E}_0}$ and $\Pi_{BC}^{\mathbf{E}_0}$. The set of updates on $\Pi_{AB}^{\mathbf{E}_1}$ which are propagation independent with respect to $\Pi_{BC}^{\mathbf{E}_1}$ is not independent of the particular state of $\Pi_{BC}^{\mathbf{E}_1}$. For example, consider the two states $M_{10} = \{R(\mathrm{a}_1, \mathrm{b}_1, \mathrm{c}_1), R(\mathrm{a}_2, \mathrm{b}_2, \mathrm{c}_1)\}$ and $M_{10'} = \{R(\mathrm{a}_1, \mathrm{b}_1, \mathrm{c}_1), R(\mathrm{a}_2, \mathrm{b}_2, \mathrm{c}_2)\}$ in $\mathsf{LDB}(\mathbf{E}_1)$. Then $\pi_{AB}^{\mathbf{E}_1}(M_{10}) = \pi_{AB}^{\mathbf{E}_1}(M_{10'}) = \{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_2, \mathrm{b}_2)\}$. The view update which replaces $\{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_2, \mathrm{b}_2)\}$ with $\{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_1, \mathrm{b}_2)\}$ on $\Pi_{AB}^{\mathbf{E}_1}$ has a reflection which keeps the state of $\Pi_{BC}^{\mathbf{E}_1}$ constant from M_{10} but not from $M_{10'}$. Thus, this view update does not exhibit context independence.

The key difference between $\mathbf{E}_0$ and $\mathbf{E}_1$ is that in the former the governing FDs embed into the views, while in the latter they do not. As first presented in [22, Thm. 2], and in a much more general context in [11, Prop. 2.17], the updates which are possible on a given view $\mathit{\Gamma}$ while keeping a second view $\mathit{\Gamma}$ constant are independent of the state of $\mathit{\Gamma}'$ iff the constraints of the main schema embed into the two views.

The conventional wisdom is that context-independent updates to views such as $\mathit{\Pi}^{\mathbf{E}_1}_{AB}$ are not possible, because checking the FD $A \to C$ requires access to both views. While this is true if one insists upon characterizing the allowable view updates as those which keep a meet view constant, it is nevertheless possible to support weaker, but still very useful, forms of context and propagation independence in such settings. It is the main goal of this paper to develop such notions of independence.

Given $N_1 \in \mathsf{LDB}(\mathbf{E}_1^{AB})$, let $\pi_B(N_1)$ denote $\{\mathrm{b} \mid (\exists t \in N_1)(t[B] = b\}$, that is, the set of all values for attribute B which occur in some tuple of N_1, and let $\equiv^{N_1}_{\langle B,A\rangle}$ denote the equivalence relation on $\pi_B(N_1)$ which identifies two B-values iff they share a common value for attribute A. Thus, $\mathrm{b}_1 \equiv^{N_1}_{\langle B,A\rangle} \mathrm{b}_2$ iff there are tuples $t_1, t_2 \in N_1$ with $t_1[B] = \mathrm{b}_1$, $t_2[B] = \mathrm{b}_2$, and $t_1[A] = t_2[A]$. It is not difficult to see that any view update (N_1, N_1') to $\mathit{\Pi}^{\mathbf{E}_1}_{AB}$ for which $\pi_B(N_1) = \pi_B(N_1')$ and for which $\equiv^{N_1'}_{\langle B,A\rangle} \subseteq \equiv^{N_1}_{\langle B,A\rangle}$ cannot lead to a violation of the FD $A \to C$ as long as the state of $\mathit{\Pi}^{\mathbf{E}_1}_{BC}$ is held constant in the reflection. For example, if the current state of $\mathit{\Pi}^{\mathbf{E}_1}_{AB}$ is $N_{11} = \{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_2, \mathrm{b}_2), R_{AB}(\mathrm{a}_2, \mathrm{b}_3)\}$, then the update to the new state $N_{11'} = \{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_2, \mathrm{b}_2), R_{AB}(\mathrm{a}_3, \mathrm{b}_3)\}$, as well as to the new state $N_{11''} = \{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_3, \mathrm{b}_2), R_{AB}(\mathrm{a}_3, \mathrm{b}_3)\}$, cannot possibly result in a violation of $A \to C$, as long as the state of $\mathit{\Pi}^{\mathbf{E}_1}_{BC}$ is held constant, regardless of what that state is. In other words, limiting the view updates to those which satisfy these properties results in a strategy which is both context and propagation independent. A similar argument holds for updates on $\mathit{\Pi}^{\mathbf{E}_1}_{BC}$. For any $N_2 \in \mathsf{LDB}(\mathbf{E}_1^{BC})$, let $\equiv^{N_2}_{\langle B,C\rangle}$ denote the equivalence relation on $\pi_B(N_2)$ which identifies two B-values if they share a common value for attribute C. Now, any view update (N_2, N_2') with $\pi_B(N_2) = \pi_B(N_2')$ and for which $\equiv^{N_2}_{\langle B,C\rangle} \subseteq \equiv^{N_2'}_{\langle B,C\rangle}$ has a reflection with constant $\mathit{\Pi}^{\mathbf{E}_1}_{AB}$ which is both context and propagation independent. Furthermore, these updates may be made to $\mathit{\Pi}^{\mathbf{E}_1}_{AB}$ and $\mathit{\Pi}^{\mathbf{E}_1}_{BC}$ independently of each other without violating any integrity constraints. The compromise, relative to that of the views of $\mathbf{E}_0$, is that the allowable updates are with respect to a given initial context $(N_1, N_2) \in \mathsf{LDB}(\mathbf{E}_1^{AB}) \times \mathsf{LDB}(\mathbf{E}_1^{BC})$. In the case of $\mathbf{E}_0$, the identification of independent updates to $\mathit{\Pi}^{\mathbf{E}_0}_{AB}$ requires no knowledge of the state of $\mathit{\Pi}^{\mathbf{E}_0}_{BC}$. In the case of $\mathbf{E}_1$, knowledge that the state of each view is the result of context-independent updates from a consistent initial state is necessary. Furthermore, each view must know its image of that initial state. Thus, $\mathit{\Pi}^{\mathbf{E}_1}_{AB}$ must know N_1 and $\mathit{\Pi}^{\mathbf{E}_1}_{BC}$ must know N_2 (but $\mathit{\Pi}^{\mathbf{E}_1}_{AB}$ need not know N_2 and $\mathit{\Pi}^{\mathbf{E}_1}_{BC}$ need not know N_1).

There is a further improvement which may be made. Note that the set of allowable updates in this example is not symmetric. For example, updating the state of $\varPi^{\mathbf{E}_1}_{AB}$ from N_{11} to $N_{11'}$ is always admissible, but the reverse, from $N_{11'}$ to N_{11} is not, since the latter may lead to a violation of $A \to C$ for certain compatible states of $\varPi^{\mathbf{E}_1}_{BC}$. Nevertheless, for any $N_{12} \in \mathsf{LDB}(\mathbf{E}^{BC}_1)$ which is compatible with N_{11} in the sense that they arise from a common $M \in \mathsf{LDB}(\mathbf{E}_1)$, this update is reversible. In fact, it remains reversible if the only updates to $\varPi^{\mathbf{E}_1}_{BC}$ are those described above, with $\equiv^{N_2}_{\langle B,C\rangle} \subseteq \equiv^{N'_2}_{\langle B,C\rangle}$. and $\pi_B(N_2) = \pi_B(N'_2)$.

A similar solution is applicable when normalization replaces two-way inclusion dependencies with simple foreign-key dependencies. That example is developed in detail in Examples 3.4.

The main goal of this paper is to place the ideas illustrated by these examples on firm theoretical footing. In contrast to the constant-complement theory, which looks primarily at how a single view may be updated while keeping a second view constant, the focus here is upon how two views may be updated independently. Furthermore, while the work is primarily within the setting of just two views, the long-term goal is nevertheless to address the situation in which there is a larger set of views, as often occurs in the application settings identified above. To this end, the main results are developed without requiring that the views be complementary. Interestingly, complementation does not appear to be a central issue and their is little if any compromise involved.

2 Schemata and Views in a General Framework

Although most of the examples are based upon the relational model, the results of this paper depend only upon the set-theoretic properties of database schemata and views. As such, the underlying framework is basically that employed in the classical papers [6] and [5]. The purpose of this section is to present the essential ideas of that framework in a succinct fashion. The terminology and notation is closest to that employed in [11], to which the reader is referred for details.

Definition 2.1 (Database schemata and morphisms). A database schema $\mathbf{D}$ is modelled completely by its set $\mathsf{LDB}(\mathbf{D})$ of *legal databases* or *states.* A morphism $f : \mathbf{D}_1 \to \mathbf{D}_2$ of database schemata is represented completely by its underlying function $f : \mathsf{LDB}(\mathbf{D}_1) \to \mathsf{LDB}(\mathbf{D}_2)$. Since no confusion can result, the morphism and its underlying function will be represented by the same symbol. Of course, schemata may have further structure (such as relational structure), and morphisms may be defined by the relational algebra or calculus, but for this work, it is only the underlying sets and functions which are of formal importance.

Definition 2.2 (Views). A *view* $\varGamma = (\mathbf{V}, \gamma)$ of the schema $\mathbf{D}$ is given by a database schema $\mathbf{V}$ together with a morphism $\gamma : \mathbf{D} \to \mathbf{V}$ whose underlying function $\gamma : \mathsf{LDB}(\mathbf{D}) \to \mathsf{LDB}(\mathbf{V})$ is surjective. In a view $\varGamma$, the state of its schema $\mathbf{V}$ is always determined completely by the state of the main schema $\mathbf{D}$.

The *congruence* $\mathsf{Congr}(\varGamma)$ of the view $\varGamma$ is the equivalence relation on $\mathsf{LDB}(\mathbf{D})$ given by $\{(M_1, M_2) \in \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D}) \mid \gamma(M_1) = \gamma(M_2)\}$. Let

$\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ be views of the schema $\mathbf{D}$. Write $\Gamma_1 \preceq_{\mathbf{D}} \Gamma_2$ just in case $\mathsf{Congr}(\Gamma_2) \subseteq \mathsf{Congr}(\Gamma_1)$, that is, just in case Γ_2 preserves at least as much information about the state of $\mathbf{D}$ as does Γ_1. The two views Γ_1 and Γ_2 are said to be *isomorphic* if $\mathsf{Congr}(\Gamma_1) = \mathsf{Congr}(\Gamma_2)$; i.e., if $\Gamma_2 \preceq_{\mathbf{D}} \Gamma_1 \preceq_{\mathbf{D}} \Gamma_2$. It is easy to see that $\preceq_{\mathbf{D}}$ is a preorder on the collection of all views of $\mathbf{D}$ and a partial order on the congruences (i.e., on the views up to isomorphism).

A *partition* on $\mathsf{LDB}(\mathbf{D})$ is given by a set P of nonempty subsets of $\mathsf{LDB}(\mathbf{D})$ with the property that each $M \in \mathsf{LDB}(\mathbf{D})$ is in exactly one element of P. Each member of P is called a *block* of the partition. The partition on $\mathsf{LDB}(\mathbf{D})$ *induced by* $\mathsf{Congr}(\Gamma)$ has M_1 and M_2 in the same block iff $\gamma(M_1) = \gamma(M_2)$. Thus, a congruence on $\mathsf{LDB}(\mathbf{D})$ may be represented by the partition which it induces [20, Sec. 1]. The partition of $\mathsf{LDB}(\mathbf{D})$ induced by $\mathsf{Congr}(\Gamma)$ is denoted $\mathsf{Partition}(\mathsf{Congr}(\Gamma))$.

Definition 2.3 (Relativized views). Let $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ be views of the schema $\mathbf{D}$. If $\Gamma_1 \preceq_{\mathbf{D}} \Gamma_2$, then Γ_2 may be *relativized* to a view of $\mathbf{V}_1$. More specifically, the function $\lambda\langle \Gamma_1, \Gamma_2 \rangle : \mathsf{LDB}(\mathbf{V}_1) \to \mathsf{LDB}(\mathbf{V}_2)$ is defined via the view congruences by sending a block β of $\mathsf{Partition}(\mathsf{Congr}(\Gamma_1))$ to the block of $\mathsf{Partition}(\mathsf{Congr}(\Gamma_2))$ which contains β. For example, using views of the $\mathbf{E}_0$ introduced in Sec. 1, $\lambda\langle \Pi^{\mathbf{E}_0}_{AB}, \Pi^{\mathbf{E}_0}_{B} \rangle$ sends a state in $\mathsf{LDB}(\Pi^{\mathbf{E}_0}_{AB})$, i.e., a relation for $R_{AB}[AB]$, to its projection on B. In terms of blocks of the equivalence relations, it sends a block β of $\mathsf{Partition}(\mathsf{Congr}(\Gamma))$ consisting of all states with the same projection onto AB, to the block of $\mathsf{Partition}(\mathsf{Congr}(\Pi^{\mathbf{E}_0}_{B}))$ with the projection onto attribute B of the elements of β.

Definition 2.4 (The lattice structure and meets of views). It is a classical result [20, Thm. 5] that the set of all congruences on a set (and hence the set of all views on a database schema) forms a bounded complete lattice (see [9, 2.2 and 2.4] for definitions) under the order induced by $\preceq_{\mathbf{D}}$. More precisely, let Γ_1 and Γ_2 be any views of the schema $\mathbf{D}$. The join will not be used in this work and so not considered further. More important is the meet $\Gamma_1 \wedge \Gamma_2 = (\mathbf{V}_1 \wedge \mathbf{V}_2, \gamma_1 \wedge \gamma_2)$, which is represented by the intersection of all equivalence relations E on $\mathsf{LDB}(\mathbf{D})$ which satisfy $E \preceq_{\mathbf{D}} \mathsf{Congr}(\Gamma_i)$ for both $i = 1$ and $i = 2$. There is always one such equivalence relation, namely the identity, so the intersection is never over the empty set. An explicit characterization of $\mathsf{Congr}(\Gamma_1 \wedge \Gamma_2)$ may be found in [20, p. 579]. Namely, $(M, M') \in \mathsf{Congr}(\Gamma_1 \wedge \Gamma_2)$ iff there is a chain

$$(M, M_1), (M_1, M_2), \ldots, (M_{i-1}, M_i), (M_i, M_{i+1}), \ldots, (M_{k-1}, M_k), (M_k, M') \tag{cc}$$

of elements in $\mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D})$ in which the right element of a pair matches the left element of its neighbor to the right, and in which each pair is either in $\mathsf{Congr}(\Gamma_1)$ or else in $\mathsf{Congr}(\Gamma_2)$.

While the join of two relational schemata always has a natural representation as a relational schema [14, Def. 3.4], the same cannot be said of the meet. Of course, it always has an abstract representation as a congruence on the states of the main schema, and in some examples, it does have a simple representation.

For example, in the context of the schema $\mathbf{E}_0$ of Sec. 1, $\Pi^{\mathbf{E}_0}_{AB} \wedge \Pi^{\mathbf{E}_0}_{BC}$ is represented by the view $\Pi^{\mathbf{E}_0}_{B}$ [11, Prop. 2.17].

The greatest view is the identity view, which has the obvious definition and which will not be considered further in this work. The least view is the *zero view*, denoted $\mathsf{ZView}_{\mathbf{D}}$, and has $\mathsf{Congr}(\mathsf{ZView}_{\mathbf{D}}) = \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D})$. It is a trivial view in that it retains no information about the state of $\mathbf{D}$; its morphism $\mathsf{ZMor}_{\mathbf{D}}$ sends every state of $\mathsf{LDB}(\mathbf{D})$ to the same, single state of the view schema.

Definition 2.5 (Commuting congruences). There is a condition which simplifies the description of the meet given in (cc) of Definition 2.4 above. The pair $\{\Gamma_1 = (\mathbf{V}_1, \gamma_1), \Gamma_2 = (\mathbf{V}_2, \gamma_2)\}$ of views is said to have *commuting congruences* if the composition of their congruences is commutative; that is, if $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) = \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1)$. In this case, the characterization (cc) may be simplified considerably. Namely, $(M, M') \in \mathsf{Congr}(\Gamma_1 \wedge \Gamma_2)$ iff there is an $M'' \in \mathsf{LDB}(\mathbf{D})$ such that $(M, M'') \in \mathsf{Congr}(\Gamma_1)$ and $(M'', M') \in \mathsf{Congr}(\Gamma_2)$ (or, equivalently, iff there is an $M'' \in \mathsf{LDB}(\mathbf{D})$ such that $(M, M'') \in \mathsf{Congr}(\Gamma_2)$ and $(M'', M') \in \mathsf{Congr}(\Gamma_1)$) [20, Sec. 8].

Definition 2.6 (Complementary views). The pair $\{\Gamma_1 = (\mathbf{V}_1, \gamma_1), \Gamma_2 = (\mathbf{V}_2, \gamma_2)\}$ of views of $\mathbf{D}$ is called *complementary* if the *decomposition morphism* $\gamma_1 \times \gamma_2 : \mathsf{LDB}(\mathbf{D}) \to \mathsf{LDB}(\mathbf{V}_1) \times \mathsf{LDB}(\mathbf{V}_2)$ given on elements by $M \mapsto (\gamma_1(M), \gamma_2(M))$ is injective. In earlier work, particularly [11], fundamental results were obtained for pairs of views which are both complementary and which have commuting congruences. Such pairs are called *meet complementary.* In this work, the property of being complementary will not be of central importance, but it will still be mentioned in some discussion of the results.

Definition 2.7 (Updates and Reflections). An *update* on the schema $\mathbf{D}$ is just a pair $(M_1, M_2) \in \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D})$. Think of M_1 as the state before the update operation and M_2 as the state afterwards. The set of all updates on $\mathbf{D}$ is denoted $\mathsf{Updates}(\mathbf{D})$.

Given a view $\Gamma = (\mathbf{V}, \gamma)$ of $\mathbf{D}$ and an update $u = (N_1, N_2) \in \mathsf{Updates}(\mathbf{V})$, a *reflection* (or *translation*) *of* u *along* Γ is a $u' = (M_1, M_2) \in \mathsf{Updates}(\mathbf{D})$ with the property that $\gamma(M_i) = N_i$ for $i \in \{1, 2\}$. In this case, u' is also called a reflection (or translation) of u *for* M_1 along Γ. The set of all reflections of u along Γ is denoted $\mathsf{Reflections}_{\Gamma}\langle u \rangle$.

3 Basic Theory of Independent Update Strategies

In this section, the central ideas surrounding independent update strategies are developed. Some of these, particularly those involving commuting congruences, have already been developed in part in the context of complementary pairs [11]. However, the focus here is not at all upon complements. Indeed, the assumption that the views under consideration are complementary is never made. Furthermore, while the emphasis in [11] is upon the constant-complement update strategy in the presence of meet complements, the main focus here is upon situations in which the meet property (i.e., commuting congruences) fails to hold.

This presentation is independent of [11], and does not require knowledge of the specific results of that paper.

Notation 3.1 (Running schema and views). Throughout this section, unless stated specifically to the contrary, take $\mathbf{D}$ to be a database schema and $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ to be views of $\mathbf{D}$. Γ_1 and Γ_2 need not be complements of each other.

Definition 3.2 (Updates relative to a second view). The goal is to identify properties on subsets of $\mathsf{Updates}(\mathbf{V}_1)$ which characterize useful yet independent update strategies. To this end, there are three distinct notions of independence which are of importance. In that which follows, let $u = (N, N') \in \mathsf{Updates}(\mathbf{V}_1)$, and define $\mathsf{Reflections}_{\Gamma_1|\Gamma_2}\langle u\rangle$ to be the subset of $\mathsf{Reflections}_{\Gamma_1}\langle u\rangle$ which keeps the state of Γ_2 constant. More precisely, $\mathsf{Reflections}_{\Gamma_1|\Gamma_2}\langle u\rangle = \{(M_1, M_2) \in \mathsf{Reflections}_{\Gamma_1}\langle u\rangle \mid (M_1, M_2) \in \mathsf{Congr}(\Gamma_2)\}$.

(a) Call u *somewhere Γ_2-independent* if for some $M_1 \in \mathsf{LDB}(\mathbf{D})$ with $\gamma_1(M_1) = N$, there is an $M_2 \in \mathsf{LDB}(\mathbf{D})$ with the property that $(M_1, M_2) \in \mathsf{Reflections}_{\Gamma_1|\Gamma_2}\langle u\rangle$. The set of all somewhere Γ_2-independent updates on Γ_1 is denoted $\mathsf{IndUpd}_{\exists}\langle\Gamma_1|\Gamma_2\rangle$.

Thus, u is somewhere Γ_2-independent if the update may be made for some states of the view Γ_2, but not necessarily all. The update $(\{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_2, \mathrm{b}_2)\}, \{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_1, \mathrm{b}_2)\})$ on $\Pi^{\mathbf{E}_1}_{AB}$ of Sec. 1 is an example which is somewhere $\Pi^{\mathbf{E}_1}_{BC}$-independent. It is not, however, everywhere independent, since there states of the view $\Pi^{\mathbf{E}_1}_{BC}$, such as $\{R_{BC}(\mathrm{b}_1, \mathrm{c}_1), R_{BC}(\mathrm{b}_2, \mathrm{c}_2)\}$, for which it cannot be realized without changing that state.

(b) Call u *everywhere Γ_2-independent* if for every $M_1 \in \mathsf{LDB}(\mathbf{D})$ with $\gamma_1(M_1) = N$, there is an $M_2 \in \mathsf{LDB}(\mathbf{D})$ with the property that $(M_1, M_2) \in \mathsf{Reflections}_{\Gamma_1|\Gamma_2}\langle u\rangle$. The set of all everywhere Γ_2-independent updates on Γ_1 is denoted $\mathsf{IndUpd}_{\forall}\langle\Gamma_1|\Gamma_2\rangle$).

The update $(\{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_2, \mathrm{b}_2)\}, \{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_3, \mathrm{b}_2)\})$ on $\Pi^{\mathbf{E}_1}_{AB}$ is an example which is everywhere $\Pi^{\mathbf{E}_1}_{BC}$-independent.

The third notion of independence characterizes independence in the situation when the congruences of Γ_1 and Γ_2 commute, and so is closely tied to the theory of constant-complement updates as presented in [11].

(c) Call u *meetwise Γ_2-independent* if $\lambda\langle\Gamma_1, \Gamma_1 \wedge \Gamma_2\rangle(N) = \lambda\langle\Gamma_1, \Gamma_1 \wedge \Gamma_2\rangle(N')$. The set of all meetwise Γ_2-independent updates on Γ_1 is denoted $\mathsf{IndUpd}_{\wedge}\langle\Gamma_1|\Gamma_2\rangle$.

For each of these three notions, there is a corresponding definition of reflected updates. Specifically, define

$$\begin{aligned}\mathsf{ReflIndUpd}_{\exists}\langle\Gamma_1|\Gamma_2\rangle &= \{\mathsf{Reflections}_{\Gamma_1|\Gamma_2}\langle u\rangle \mid u \in \mathsf{IndUpd}_{\exists}\langle\Gamma_1|\Gamma_2\rangle\},\\ \mathsf{ReflIndUpd}_{\forall}\langle\Gamma_1|\Gamma_2\rangle &= \{\mathsf{Reflections}_{\Gamma_1|\Gamma_2}\langle u\rangle \mid u \in \mathsf{IndUpd}_{\forall}\langle\Gamma_1|\Gamma_2\rangle\},\\ \text{and } \mathsf{ReflIndUpd}_{\wedge}\langle\Gamma_1|\Gamma_2\rangle &= \{\mathsf{Reflections}_{\Gamma_1|\Gamma_2}\langle u\rangle \mid u \in \mathsf{IndUpd}_{\wedge}\langle\Gamma_1|\Gamma_2\rangle\}.\end{aligned}$$

In these definitions, there is no assumption that Γ_1 and Γ_2 be complements. However, if they are complements, then for each $u = \langle N, N'\rangle \in \mathsf{IndUpd}_x\langle\Gamma_1|\Gamma_2\rangle$ with $x \in \{\exists, \forall, \wedge\}$, and each $M \in \gamma_1^{-1}(N)$, there is at most one $M' \in \mathsf{LDB}(\mathbf{D})$ with

$(M, M') \in \mathsf{ReflIndUpd}_x\langle\Gamma_1|\Gamma_2\rangle$, which given by $(\gamma_1 \times \gamma_2)^{-1}(N', \gamma_2(M))$ whenever it exists (see Definition 2.6).

All three notions of Γ_2-independence recapture locality of effect, as defined in Sec. 1. While only everywhere Γ_2 independence recaptures context independence, the other two provide crucial insights into what can go wrong and how things can be extended. As a first step, the question of whether or not these are equivalence relations is examined.

Observation 3.3 (Reflexivity and transitivity)

(a) *Each of* $\mathsf{IndUpd}_\wedge\langle\Gamma_1|\Gamma_2\rangle$, $\mathsf{IndUpd}_\exists\langle\Gamma_1|\Gamma_2\rangle$, $\mathsf{ReflIndUpd}_\wedge\langle\Gamma_1|\Gamma_2\rangle$, *and* $\mathsf{ReflIndUpd}_\exists\langle\Gamma_1|\Gamma_2\rangle$ *is an equivalence relations.*

(b) $\mathsf{IndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$ *and* $\mathsf{ReflIndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$ *are reflexive and transitive, but not necessarily symmetric. Thus, they need not be equivalence relations.*

Proof All of the "positive" conditions are routine verifications, which are left to the reader. That $\mathsf{IndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$ and $\mathsf{ReflIndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$ need not be symmetric is illustrated in Examples 3.4, immediately below. □

Examples 3.4 (Non-reversible independent updates). To illustrate the idea of non-reversible updates, consider the schema $\mathbf{E}_2$ with two relation symbols $R_{AB}[AB]$ and $R_{BC}[BC]$. The latter relation is governed by the FD $B \to C$, and, in addition, the two relations are connected via the foreign-key dependency $R_{AB}[B] \subseteq R_{BC}[B]$. Define $\Pi_{AB}^{\mathbf{E}_2} = (\mathbf{E}_2^{AB}, \pi_{AB}^{\mathbf{E}_2})$ and $\Pi_{BC}^{\mathbf{E}_2} = (\mathbf{E}_2^{BC}, \pi_{BC}^{\mathbf{E}_2})$ as the views which preserve $R_{AB}[AB]$ and $R_{BC}[BC]$, respectively, and for $N \in \mathsf{LDB}(\mathbf{E}_2^{AB})$ or $N \in \mathsf{LDB}(\mathbf{E}_2^{BC})$, let $\pi_B(N)$ denote $\{\mathrm{b} \mid (\exists t \in N)(t[B] = b\}$.

It is easy to see that $\mathsf{IndUpd}_\forall\langle\Pi_{AB}^{\mathbf{E}_2}|\Pi_{BC}^{\mathbf{E}_2}\rangle$ is the set of all updates $(N, N') \in \mathsf{Updates}(\Pi_{AB}^{\mathbf{E}_2})$ for which $\pi_B(N') \subseteq \pi_B(N)$. A tuple of the form $R_{AB}(\mathrm{a}, \mathrm{b})$ may always be deleted, even if there is no other tuple of the form $R_{AB}(x, \mathrm{b})$, but a tuple of the form $R_{AB}(\mathrm{a}, \mathrm{b})$ may not be added if there is not already another of the form $R_{AB}(x, \mathrm{b})$. Thus, if $R_{AB}(\mathrm{a}, \mathrm{b})$ is deleted, it may not be reinserted. For $\mathsf{IndUpd}_\forall\langle\Pi_{AB}^{\mathbf{E}_2}|\Pi_{BC}^{\mathbf{E}_2}\rangle$, the situation is reversed; $(N, N') \in \mathsf{IndUpd}_\forall\langle\Pi_{BC}^{\mathbf{E}_2}|\Pi_{AB}^{\mathbf{E}_2}\rangle$ iff $\pi_B(N) \subseteq \pi_B(N')$. A tuple of the form $R_{BC}(\mathrm{b}, \mathrm{c})$ may always be inserted, but not deleted unless there is another tuple of the form $R_{BC}(\mathrm{b}, x)$. Hence, neither $\mathsf{IndUpd}_\forall\langle\Pi_{AB}^{\mathbf{E}_3}|\Pi_{BC}^{\mathbf{E}_2}\rangle$ nor $\mathsf{IndUpd}_\forall\langle\Pi_{BC}^{\mathbf{E}_3}|\Pi_{AB}^{\mathbf{E}_2}\rangle$ is symmetric.

A similar situation governs the example surrounding $\mathbf{E}_1$ of Sec. 1. A view update $(N, N') \in \mathsf{IndUpd}_\forall\langle\Pi_{AB}^{\mathbf{E}_1}|\Pi_{BC}^{\mathbf{E}_1}\rangle$ is allowed if $\pi_B(N) = \pi_B(N')$ and $\equiv_{\langle B,A\rangle}^{N'} \subseteq \equiv_{\langle B,A\rangle}^{N}$, but not if $\equiv_{\langle B,A\rangle}^{N'} \not\subseteq \equiv_{\langle B,A\rangle}^{N}$. Thus, a view update of the form $(\{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_2, \mathrm{b}_2)\}, \{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_1, \mathrm{b}_2)\})$ is not allowed. An analogous condition holds for $\mathsf{IndUpd}_\forall\langle\Pi_{BC}^{\mathbf{E}_1}|\Pi_{AB}^{\mathbf{E}_1}\rangle$.

These examples suggest the way to extend the notion of independent update. Return to $\mathbf{E}_2$ and its views. Suppose that the state of the schema $\mathbf{E}_2$ is $M_{21} = \{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_2, \mathrm{b}_2), R_{BC}(\mathrm{b}_1, \mathrm{c}_1), R_{BC}(\mathrm{b}_2, \mathrm{c}_2)\}$. Then the update $u_{21} = (\{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_2, \mathrm{b}_2)\}, \{R_{AB}(\mathrm{a}_1, \mathrm{b}_1)\})$ on $\Pi_{AB}^{\mathbf{E}_2}$ is in $\mathsf{IndUpd}_\forall\langle\Pi_{AB}^{\mathbf{E}_2}|\Pi_{BC}^{\mathbf{E}_2}\rangle$ but the reverse update $u'_{21} = (\{R_{AB}(\mathrm{a}_1, \mathrm{b}_1)\}, \{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_2, \mathrm{b}_2)\})$ is not. However, if it is known that the state of $\Pi_{BC}^{\mathbf{E}_2}$ did not change after the execution of u_{21}, then a subsequent execution of u'_{21} is indeed possible while keeping $\mathit{\Pi}^{\mathbf{E}_2}_{BC}$ constant. Even stronger, if updates in $\mathsf{IndUpd}_{\forall}\langle \mathit{\Pi}^{\mathbf{E}_2}_{BC} | \mathit{\Pi}^{\mathbf{E}_2}_{AB}\rangle$ are applied, this condition nevertheless continues to hold. It is only if an update to $\mathit{\Pi}^{\mathbf{E}_2}_{BC}$ is applied which removes elements from $\pi_B(\{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_2, \mathrm{b}_2)\}) = \{\mathrm{b}_1, \mathrm{b}_2\}$ that the update u_{21} may become irreversible. The strategy is that non-reversible updates in $\mathsf{IndUpd}_{\forall}\langle \mathit{\Gamma}_1 | \mathit{\Gamma}_2\rangle$ may in fact be reversed provided that the only updates to $\mathit{\Gamma}_2$ which are allowed are those in $\mathsf{IndUpd}_{\forall}\langle \mathit{\Gamma}_2 | \mathit{\Gamma}_1\rangle$ and their reversals, all within the context of a given initial state. A systematic development of these ideas constitutes the remainder of this paper.

Proposition 3.5 (Comparison of the three notions of independent updates). $\mathsf{IndUpd}_{\forall}\langle \mathit{\Gamma}_1 | \mathit{\Gamma}_2\rangle \subseteq \mathsf{IndUpd}_{\exists}\langle \mathit{\Gamma}_1 | \mathit{\Gamma}_2\rangle \subseteq \mathsf{IndUpd}_{\wedge}\langle \mathit{\Gamma}_1 | \mathit{\Gamma}_2\rangle$*. Furthermore, their exist examples for which these inclusions are proper.*

Proof. That $\mathsf{IndUpd}_{\forall}\langle \mathit{\Gamma}_1 | \mathit{\Gamma}_2\rangle \subseteq \mathsf{IndUpd}_{\exists}\langle \mathit{\Gamma}_1 | \mathit{\Gamma}_2\rangle$ is immediate, and an example for which the inclusion is proper is given by $\mathbf{E}_1$ and its views in Sec. 1.

It is also easy to see that $\mathsf{IndUpd}_{\exists}\langle \mathit{\Gamma}_1 | \mathit{\Gamma}_2\rangle \subseteq \mathsf{IndUpd}_{\wedge}\langle \mathit{\Gamma}_1 | \mathit{\Gamma}_2\rangle$, since if $(N, N') \in \mathsf{IndUpd}_{\exists}\langle \mathit{\Gamma}_1 | \mathit{\Gamma}_2\rangle$, then by definition there is a pair $(M, M') \in \mathsf{Reflections}_{\mathit{\Gamma}_1|\mathit{\Gamma}_2}\langle (N, N')\rangle$, and $(M, M') \in \mathsf{Congr}(\mathit{\Gamma}_2)$, since the update holds $\mathit{\Gamma}_2$ constant. Then, since $\mathsf{Congr}(\mathit{\Gamma}_2) \subseteq \mathsf{Congr}(\mathit{\Gamma}_1 \wedge \mathit{\Gamma}_2)$, the results follows. For an example in which this inclusion is proper, let $\mathbf{E}_3$ have five states: $\mathsf{LDB}(\mathbf{E}_3) = \{a, b, c, d, e\}$. Let $\mathit{\Omega}_{31} = (\mathbf{V}_{31}, \omega_{31})$ be the view with $\mathsf{Partition}(\mathsf{Congr}(\mathit{\Omega}_{31})) = \{\{a, b\}, \{c, d\}, \{e\}\}$ and let $\mathit{\Omega}_{32} = (\mathbf{V}_{32}, \omega_{32})$ have $\mathsf{Partition}(\mathsf{Congr}(\mathit{\Omega}_{32})) = \{\{a\}, \{b, c\}, \{d, e\}\}$. It is easy to show that $\mathsf{Partition}(\mathsf{Congr}(\omega_{31} \wedge \omega_{32})) = \{a, b, c, d, e\}$, i.e., it is defined by $\mathsf{ZView}_{\mathbf{E}_3}$. This means that $\mathsf{IndUpd}_{\wedge}\langle \mathit{\Omega}_{31} | \mathit{\Omega}_{32}\rangle = \mathsf{LDB}(\mathbf{E}_3) \times \mathsf{LDB}(\mathbf{E}_3)$; i.e., any update is allowed. However, the update $(\{a, b\}, \{e\})$ is not in $\mathsf{IndUpd}_{\exists}\langle \mathit{\Omega}_{31} | \mathit{\Omega}_{32}\rangle$, and so $\mathsf{IndUpd}_{\wedge}\langle \mathit{\Omega}_{31} | \mathit{\Omega}_{32}\rangle$ is a proper subset of it. □

Definition 3.6 (Compatible pairs and independent update pairs). A *compatible pair* for $\{\mathit{\Gamma}_1, \mathit{\Gamma}_2\}$ is an $(N_1, N_2) \in \mathsf{LDB}(\mathbf{V}_1) \times \mathsf{LDB}(\mathbf{V}_2)$ which arises from some $M \in \mathsf{LDB}(\mathbf{D})$. If $\{\mathit{\Gamma}_1, \mathit{\Gamma}_2\}$ forms a complementary pair, then there is at most one compatible pair associated with each $M \in \mathsf{LDB}(\mathbf{D})$; namely $(\gamma_1 \wedge \gamma_2)^{-1}(N_1, N_2)$ when it exists. However, in the more general context, there may be many, since $\gamma_1 \times \gamma_2$ need not be injective. Formally, define $\mathsf{Compat}\langle \mathit{\Gamma}_1; \mathit{\Gamma}_2\rangle = \{(N_1, N_2) \in \mathsf{LDB}(\mathbf{V}_1) \times \mathsf{LDB}(\mathbf{V}_2) \mid (\exists M \in \mathsf{LDB}(\mathbf{D}))(\forall i \in \{1, 2\})(\gamma_i(M) = N_i)\}$.

An *independent update pair* is a pair of updates $(u_1, u_2) \in \mathsf{Updates}(\mathbf{V}_1) \times \mathsf{Updates}(\mathbf{V}_2)$ which may be executed independently of one another. Formally, define $\mathsf{IndUpd}_{\forall}\langle \mathit{\Gamma}_1 \| \mathit{\Gamma}_2\rangle = \{(N_1, N_2), (N'_1, N'_2) \mid (N_1, N_2) \in \mathsf{Compat}\langle \mathit{\Gamma}_1; \mathit{\Gamma}_2\rangle$ and $(N_1, N'_1) \in \mathsf{IndUpd}_{\forall}\langle \mathit{\Gamma}_1 | \mathit{\Gamma}_2\rangle$ and $(N_2, N'_2) \in \mathsf{IndUpd}_{\forall}\langle \mathit{\Gamma}_2 | \mathit{\Gamma}_1\rangle\}$.

Updates in $\mathsf{IndUpd}_{\forall}\langle \mathit{\Gamma}_1 \| \mathit{\Gamma}_2\rangle$ may be performed individually as updates in $\mathsf{IndUpd}_{\forall}\langle \mathit{\Gamma}_1 | \mathit{\Gamma}_2\rangle$ and $\mathsf{IndUpd}_{\forall}\langle \mathit{\Gamma}_2 | \mathit{\Gamma}_1\rangle$, as well as concurrently, and these operations all preserve compatibility. Formally, this is expressed as follows.

Proposition 3.7 (Independent updates). *Let* $(N_1, N_2) \in \mathsf{Compat}\langle \mathit{\Gamma}_1; \mathit{\Gamma}_2\rangle$ *and let* $((N_1, N'_1), (N_2, N'_2)) \in \mathsf{IndUpd}_{\forall}\langle \mathit{\Gamma}_1 \| \mathit{\Gamma}_2\rangle$*. Then each of the pairs* $((N_1, N'_1), (N_2, N_2))$, $((N_1, N'_1), (N'_2, N'_2))$, $((N_1, N_1), (N_2, N'_2))$, *and*

$((N_1', N_1'), (N_2, N_2'))$ is in $\mathsf{IndUpd}_\forall\langle\Gamma_1\|\Gamma_2\rangle$ as well. In particular, each of (N_1', N_2'), (N_1', N_2), and (N_1', N_2') is in $\mathsf{Compat}\langle\Gamma_1;\Gamma_2\rangle$.

Proof. It suffices to equate certain elements in the description of Definition 3.6. For example, letting N_2' be N_2, which is always possible since identity updates such as (N_2, N_2) are in $\mathsf{IndUpd}_\forall\langle\Gamma_2|\Gamma_1\rangle$ regardless of the choices of Γ_1 and Γ_2, it follows that $((N_1, N_1'), (N_2, N_2)) \in \mathsf{IndUpd}_\forall\langle\Gamma_1\|\Gamma_2\rangle$. The other three cases are shown similarly. □

The next theorem provides a comprehensive characterization of the conditions for independent updates, without any requirement of complementation. The equivalence of (a), (b), and (c) has already been shown in [11, 2.14] for the special case of complementary pairs, using a different approach [11, Thm. 2.14].

Theorem 3.8 (Independence and commuting congruences). *The following conditions are equivalent.*

(a) *The pair $\{\Gamma_1, \Gamma_2\}$ has commuting congruences; i.e., $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) = \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1)$.*

(b) *For any $N_1 \in \mathsf{LDB}(\mathbf{V}_1)$ and $N_2 \in \mathsf{LDB}(\mathbf{V}_2)$, $(N_1, N_2) \in \mathsf{Compat}\langle\Gamma_1;\Gamma_2\rangle$ iff $\lambda\langle\Gamma_1, \Gamma_1 \wedge \Gamma_2\rangle(N_1) = \lambda\langle\Gamma_2, \Gamma_1 \wedge \Gamma_2\rangle(N_2)$.*

(c) *For any $N_1, N_1' \in \mathsf{LDB}(\mathbf{V}_1)$ and $N_2, N_2' \in \mathsf{LDB}(\mathbf{V}_2)$, if any three elements of the set $\{(N_1, N_2), (N_1, N_2'), (N_1', N_2), (N_1', N_2')\}$ are in $\mathsf{Compat}\langle\Gamma_1;\Gamma_2\rangle$, then so too is the fourth.*

(d) $\mathsf{IndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle = \mathsf{IndUpd}_\wedge\langle\Gamma_1|\Gamma_2\rangle$.

(e) $\mathsf{IndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle = \mathsf{IndUpd}_\exists\langle\Gamma_1|\Gamma_2\rangle$.

(f) $\mathsf{ReflIndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle = \mathsf{Congr}(\Gamma_2)$.

(g) $\mathsf{ReflIndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$ *is an equivalence relation.*

(h) $\mathsf{IndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$ *is an equivalence relation.*

Proof. ((a) ⇒ (b)): First, assume that $\lambda\langle\Gamma_1, \Gamma_1 \wedge \Gamma_2\rangle(N_1) = \lambda\langle\Gamma_2, \Gamma_1 \wedge \Gamma_2\rangle(N_2)$, and let $M_1, M_2 \in \mathsf{LDB}(\mathbf{D})$ with $\gamma_1(M_1) = N_1$ and $\gamma_2(M_2) = N_2$. Then $(\gamma_1 \wedge \gamma_2)(M_1) = (\gamma_1 \wedge \gamma_2)(M_2)$, i.e., $(M_1, M_2) \in \mathsf{Congr}(\Gamma_1 \wedge \Gamma_2)$. Using the characterization of $\mathsf{Congr}(\Gamma_1 \wedge \Gamma_2)$ for commuting congruences given in Definition 2.4, there must be an $M \in \mathsf{LDB}(\mathbf{D})$ with $(M_1, M) \in \mathsf{Congr}(\Gamma_1)$ and $(M, M_2) \in \mathsf{Congr}(\Gamma_2)$. Furthermore, $\gamma_1(M) = \gamma_1(M_1) = N_1$, and $\gamma_2(M) = \gamma_1(M_2) = N_2$, whence $(N_1, N_2) \in \mathsf{Compat}\langle\Gamma_1;\Gamma_2\rangle$. Conversely, if $(N_1, N_2) \in \mathsf{Compat}\langle\Gamma_1;\Gamma_2\rangle$, then there exists an $M \in \mathsf{LDB}(\mathbf{D})$ with $\gamma_1(M) = N_1$ and $\gamma_2(M) = N_2$. Since this M maps to a single block of $\mathsf{Partition}(\mathsf{Congr}(\Gamma_1 \wedge \Gamma_2))$, N_1 and N_2 must be associated with the same block as well.

((b) ⇒ (c)): Immediate.

((c) ⇒ (e)): Let $(N_1, N_1') \in \mathsf{IndUpd}_\exists\langle\Gamma_1|\Gamma_2\rangle$, and choose $(M_1, M_2) \in \mathsf{Reflections}_{\Gamma_1|\Gamma_2}\langle(N_1, N_2)\rangle$. Then $(N_1, \gamma_2(M_1)), (N_1', \gamma_2(M_1)) \in \mathsf{Compat}\langle\Gamma_1;\Gamma_2\rangle$. Choose $M_1' \in \mathsf{LDB}(\mathbf{D})$ with $\gamma_1(M_1') = N_1$. Then $(N_1, \gamma_2(M_1')) \in \mathsf{Compat}\langle\Gamma_1;\Gamma_2\rangle$ as well. Hence, by (c), $(N_1', \gamma_2(M_1')) \in \mathsf{Compat}\langle\Gamma_1;\Gamma_2\rangle$, whence $(N_1, N_1') \in \mathsf{IndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$, as required.

((e) ⇒ (f)): It is immediate that $\mathsf{ReflIndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle \subseteq \mathsf{Congr}(\Gamma_2)$. Conversely, let $(M_1, M_2) \in \mathsf{Congr}(\Gamma_2)$. Then $(\gamma_1(M_1), \gamma_1(M_2)) \in \mathsf{IndUpd}_\exists\langle\Gamma_1|\Gamma_2\rangle$, just by construction. Hence, invoking (e), $(\gamma_1(M_1), \gamma_1(M_2)) \in \mathsf{IndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$ as well, whence $(M_1, M_2) \in \mathsf{ReflIndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$ and so $\mathsf{ReflIndUpd}_\exists\langle\Gamma_1|\Gamma_2\rangle = \mathsf{ReflIndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$.

((f) $\Rightarrow$ (g)): Immediate.

((g) $\Rightarrow$ (h)): The proof is a routine verification.

((h) $\Rightarrow$ (a)): Let $(M_1, M_2) \in \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)$. Then there is an $M' \in \mathsf{LDB}(\mathbf{D})$ with $(M_1, M') \in \mathsf{Congr}(\Gamma_1)$ and $(M', M_2) \in \mathsf{Congr}(\Gamma_2)$. Since $\gamma_2(M') = \gamma_2(M_2)$ and $(M_1, M') \in \mathsf{IndUpd}_\exists\langle\Gamma_1|\Gamma_2\rangle$, it follows that $(\gamma_1(M_1), \gamma_1(M_2)) \in \mathsf{IndUpd}_\exists\langle\Gamma_1|\Gamma_2\rangle$. Now, choose any $M_1' \in \mathsf{LDB}(\mathbf{D})$ with $\gamma_1(M_1') = \gamma_1(M_1)$. Then $(M_1', M_2) = (M_1', M_1) \circ (M_1, M_2) \in \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) = \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)$, and so $(\gamma_1(M_1), \gamma_1(M_2)) \in \mathsf{IndUpd}_\exists\langle\Gamma_1|\Gamma_2\rangle$ as well. Since M_1' was arbitrary with $\gamma(M_1) = \gamma(M_1')$, it follows that $(\gamma_1(M_1), \gamma_1(M_2)) \in \mathsf{IndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$. Conversely, if $(M_1, M_2) \in \mathsf{Updates}(\mathbf{D})$ with $(\gamma_1(M_1), \gamma_1(M_2)) \in \mathsf{IndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$, then there must be an $M' \in \mathsf{LDB}(\mathbf{D})$ with $(M_1, M') \in \mathsf{Congr}(\Gamma_1)$ and $(M', M_2) \in \mathsf{Congr}(\Gamma_2)$; i.e., $(M', M_2) \in \mathsf{ReflIndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$. In other words, $(M_1, M_2) \in \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)$. Thus, $(M_1, M_2) \in \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)$ iff $(\gamma_1(M_1), \gamma_2(M_2)) \in \mathsf{IndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$. Since $\mathsf{IndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$ is assumed to be an equivalence relation, it follows easily that $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)$ must be an equivalence relation as well. Then $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) = \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1)$ follows immediately, since one is the reverse of the other; i.e., $(M_1, M_2) \in \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)$ iff $(M_2, M_1) \in \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1)$.

((a) $\Rightarrow$ (d)): Let $(N_1, N_1') \in \mathsf{IndUpd}_\wedge\langle\Gamma_1|\Gamma_2\rangle$. Then, as described in Definition 2.4, for any $(M, M') \in \mathsf{Reflections}_{\Gamma_1|\Gamma_2}\langle(N, N')\rangle$, there is a sequence $M_1, M_2, \ldots, M_k$ of elements of $\mathsf{LDB}(\mathbf{D})$ with the property that $M_1 = M$, $M_k = M'$, for each odd i, $1 \leq i \leq k$, $(M_i, M_{i+1}) \in \mathsf{Congr}(\Gamma_2)$ and for each even i, $1 \leq i \leq k$, $(M_i, M_{i+1}) \in \mathsf{Congr}(\Gamma_1)$. However, in view of condition (a), which guarantees commuting congruences, it follows also from the discussion of Definition 2.4 that k may be chosen to be 3. That is, there are $(M_1, M_2) \in \mathsf{Congr}(\Gamma_2)$ and $(M_2, M_3) \in \mathsf{Congr}(\Gamma_1)$ with $\gamma_1(M_1) = N_1$, $\gamma_1(M_3) = N_1'$, and $\gamma_2(M_2) = \gamma_2(M_3)$. Since M may be chosen arbitrarily with the property that $\gamma_1(M_1) = N$, this means in particular that $(N_1, N_1') = (\gamma_1(M_1), \gamma_1(M_2)) \in \mathsf{IndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$, as required. (That M may be chosen arbitrarily follows from the fact that the equivalence relation $\mathsf{Congr}(\Gamma_1 \wedge \Gamma_2)$ is coarser than $\mathsf{Congr}(\Gamma_2)$, and so any two elements of $\mathsf{LDB}(\mathbf{D})$ which are equivalent under $\mathsf{Congr}(\Gamma_2)$ are equivalent under $\mathsf{Congr}(\Gamma_1) \wedge \mathsf{Congr}(\Gamma_2)$ as well.)

((d) $\Rightarrow$ (e)): This follows immediately from Proposition 3.5. □

The thrust of this result is that $\mathsf{IndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$ being an equivalence relation is both a necessary and a sufficient condition for the classical characterization in terms of commuting congruences and meet dependencies (a)-(c) to hold, and under these conditions, each of the concepts of independent update $\mathsf{IndUpd}_\forall\langle\Gamma_1|\Gamma_2\rangle$, $\mathsf{IndUpd}_\exists\langle\Gamma_1|\Gamma_2\rangle$, and $\mathsf{IndUpd}_\wedge\langle\Gamma_1|\Gamma_2\rangle$ becomes equivalent to all of the others.

There is furthermore a symmetry in results (d)-(f); if Γ_1 and Γ_2 are swapped in any or all of these, the result remains valid. In particular, $\mathsf{IndUpd}_{\forall}\langle \Gamma_1 | \Gamma_2 \rangle$ is an equivalence relation iff $\mathsf{IndUpd}_{\forall}\langle \Gamma_2 | \Gamma_1 \rangle$ is. In other words, if independent updates are well behaved on Γ_1, then they are well behaved on Γ_2 as well.

The question becomes, then, how to recapture the extended updates identified in the examples of Sec. 1 and Examples 3.4. The answer is that rather than trying to avoid allowing $\mathsf{IndUpd}_{\forall}\langle \Gamma_1 | \Gamma_2 \rangle$ to become an equivalence relation (which in view of the above result would imply many other limitations), the set of allowable legal databases is trimmed so that $\mathsf{IndUpd}_{\forall}\langle \Gamma_1 | \Gamma_2 \rangle$ (and so $\mathsf{IndUpd}_{\forall}\langle \Gamma_2 | \Gamma_1 \rangle$ as well) becomes an equivalence relation on that which remains. The key idea is to start with a pair $(N_1, N_2) \in \mathsf{LDB}(\mathbf{V}_1) \times \mathsf{LDB}(\mathbf{V}_2)$, and then restrict attention to those states which can be reached from those via well-behaved updates. The formalization is as follows.

Definition 3.9 (Reachability subschemata and subviews). For $(N_1, N_2) \in \mathsf{Compat}\langle \Gamma_1; \Gamma_2 \rangle$, define $\mathsf{Reachable}_{\forall}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle =$

$$\{M \in \mathsf{LDB}(\mathbf{D}) \mid ((N_1, N_2), (\gamma_1(M), \gamma_2(M)) \in \mathsf{IndUpd}_{\forall}\langle \Gamma_1 \| \Gamma_2 \rangle)\}.$$

Thus, $\mathsf{Reachable}_{\forall}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle$ is the set of all states of $\mathbf{D}$ which can be reached via independent updates on Γ_1 and Γ_2 from a state $M_0 \in \mathsf{LDB}(\mathbf{D})$ with $\gamma_1(M_0) = N_1$ and $\gamma_2(M_0) = N_2$. If $\{\Gamma_1, \Gamma_2\}$ forms a complementary pair, then this initial M_0 is determined completely by (N_1, N_2), but it is not necessary to enforce complementation in that which follows.

A limited view based upon Γ_1, which only involves the reachable states, is defined as follows.

(a) Define $\mathsf{Restr}_{\mathbf{D}}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle$ to be the subschema of $\mathbf{D}$ with $\mathsf{LDB}(\mathsf{Restr}_{\mathbf{D}}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle) = \mathsf{Reachable}_{\forall}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle$.

Thus, $\mathsf{Restr}_{\mathbf{D}}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle$ is the schema consisting of just those states reachable from (N_1, N_2). The corresponding sets of view states are defined as follows.

(b) For $i \in \{1, 2\}$, define $\mathsf{Restr}_{\mathbf{V}_i}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle = \{\gamma_i(M) \mid M \in \mathsf{Reachable}_{\forall}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle\}$.

The corresponding view morphism is then the appropriate restriction of γ_i.

(c) For $i \in \{1, 2\}$, define the function

$$\mathsf{Restr}_{\gamma_i}\langle \gamma_1 : N_1 \parallel \gamma_2 : N_2 \rangle : \mathsf{LDB}(\mathsf{Restr}_{\mathbf{D}}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle) \to \mathsf{LDB}(\mathsf{Restr}_{\mathbf{V}_1}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle)$$

to be the restriction of γ_i to $\mathsf{LDB}(\mathsf{Restr}_{\mathbf{D}}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle)$.

Finally, the restricted view is obtained by assembling these pieces.

(d) For $i \in \{1, 2\}$, define

$$\mathsf{Restr}_{\Gamma_i}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle = (\mathsf{Restr}_{\mathbf{V}_i}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle, \mathsf{Restr}_{\gamma_i}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle)$$

to be the view of $\mathsf{Restr}_{\mathbf{D}}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle$ constructed from these.

That this view provides exactly that which is needed to support the extended and reversible set of independent updates for a pair of views is recaptured in the following.

Theorem 3.10 (The restricted view defined by a compatible pair). *The view* $\{\mathsf{Restr}_{\Gamma_1}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle, \mathsf{Restr}_{\Gamma_2}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2 \rangle\}$ *has commuting congruences with*

$$\mathsf{Restr}_{\Gamma_1}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2\rangle \wedge \mathsf{Restr}_{\Gamma_2}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2\rangle = \mathsf{ZView}_{\mathsf{Restr}_{\mathbf{D}}\langle \Gamma_1 : N_1 \parallel \Gamma_2 : N_2\rangle}.$$

Proof. There is really nothing difficult to prove; the given properties are crafted right into the definition. In particular, the meet is the zero view because the interdependence conditions which place limitations on the allowable updates are enforced by including only those states which are already compatible. □

Examples 3.11 (Independent view updates in the reachability context). Consider first the views $\Pi^{\mathbf{E}_2}_{AB}$ and $\Pi^{\mathbf{E}_2}_{BC}$ associated with the schema $\mathbf{E}_2$, as introduced in Examples 3.4. Let $(N_1, N_2) \in \mathsf{Compat}\langle \Pi^{\mathbf{E}_2}_{AB}; \Pi^{\mathbf{E}_2}_{BC}\rangle$. The key information which is used to characterize the admissible updates is found in the sets $\pi_B(N_1)$ and $\pi_B(N_2)$. Specifically, $\mathsf{LDB}(\mathsf{Restr}_{\mathbf{E}_2}\langle \Pi^{\mathbf{E}_2}_{AB} : N_1 \parallel \Pi^{\mathbf{E}_2}_{BC} : N_2\rangle)$ $= \mathsf{Reachable}_{\forall}\langle \Pi^{\mathbf{E}_2}_{AB} : N_1 \parallel \Pi^{\mathbf{E}_2}_{BC} : N_2\rangle = \{(N'_1, N'_2) \in \mathsf{LDB}(\mathbf{E}^{AB}_2) \times \mathsf{LDB}(\mathbf{E}^{BC}_2) \mid \pi_B(N'_1) \subseteq \pi_B(N_1) \text{ and } \pi_B(N_2) \subseteq \pi_B(N'_2)\}$. Thus, updates to the schemata of these two views are constrained only in that the initial projection of the relation of $R_{AB}[AB]$ onto B may not increase, and the initial projection of the relation of $R_{BC}[BC]$ may not decrease. This is far more flexible than the constant-complement solution suggested in [15, Discussion 3.1]. In that solution, in order to maintain a meet situation, a copy of the projection of $\Pi^{\mathbf{E}_2}_{BC}$ onto B must be included in the view $\Pi^{\mathbf{E}_2}_{AB}$. That limits the allowable updates to those which keep both $\pi_B(N_1)$ and $\pi_B(N_2)$ constant, a much smaller set.

Next, consider the views associated with $\mathbf{E}_1$. Here the classical constant-complement update strategy would allow no updates at all to either view. However, with the restricted views, the allowable updates are those which satisfy the conditions identified in Sec. 1. For a given $(N_1, N_2) \in \mathsf{Compat}\langle \Pi^{\mathbf{E}_1}_{AB}; \Pi^{\mathbf{E}_1}_{BC}\rangle$, using the definitions of $\equiv^{N}_{\langle X,Y\rangle}$ given in Sec. 1, $\mathsf{LDB}(\mathsf{Restr}_{\mathbf{E}_2}\langle \Pi^{\mathbf{E}_2}_{AB} : N_1 \parallel \Pi^{\mathbf{E}_2}_{BC} : N_2\rangle)$ $= \mathsf{Reachable}_{\forall}\langle \Pi^{\mathbf{E}_1}_{AB} : N_1 \parallel \Pi^{\mathbf{E}_2}_{BC} : N_2\rangle = \{(N'_1, N'_2) \in \mathsf{LDB}(\mathbf{E}^{AB}_1) \times \mathsf{LDB}(\mathbf{E}^{BC}_2) \mid \pi_B(N_1) = \pi_B(N_2) \text{ and } \equiv^{N'_1}_{\langle B,A\rangle} \subseteq \equiv^{N_1}_{\langle B,A\rangle} \text{ and } \equiv^{N_2}_{\langle B,C\rangle} \subseteq \equiv^{N'_2}_{\langle B,C\rangle}\}$. Parallel updates by the two views may reach any of these states.

The price paid for using this type of update strategy is that the constraints defining which updates are allowed must be reset every time the pair of views is updated outside of this framework. That would happen, for example, when an update not supported in the restricted strategy were necessary, and so the two views would be combined, the update performed, and then a new initial compatible pair obtained. Whether the cost of computing anew which view updates are supported every time the initial compatible pair changes is a worthwhile step depends upon the application setting, and must be left as topic for further investigation.

4 Conclusions and Further Directions

A way to handle updates on two views, without any conflict, has been presented. The approach extends the classical constant-complement strategy in two ways. First and foremost, it is not restricted to meet complements (translatable strategies in the language of [6]). Rather, it takes advantage of the fact that simultaneous updates are limited in scope, and assumes that the updates to the companion view follow the associated protocol. Second, it does not depend upon complementation in any way, and so is readily extensible to any finite number of views.

Directions for additional investigation include the following:

Extension to finite sets of views: As noted in Sec. 1, a primary motivation for this work is the modelling in the context of many views. It is therefore of primary importance to develop the details of how this approach extends to more than two views.

Integration with applications: The ideas developed here should be of great use in extending the notion of database schema components, as described in [13] and [16], as well as their applications in update via cooperation [17] and objects for transaction [15]. The next task is to examine the details of such applications.

Effective methods for identifying the restricted state set: In the approach developed in this paper, the allowable updates are defined by a starting context (the reachability subschema). It is important to identify ways to characterize and compute effectively this context for classes of views which arise in practice.

Acknowledgment. An anonymous reviewer made numerous suggestions which (hopefully) have led to a more readable presentation.

References

1. Arenas, M., Bertossi, L.E., Chomicki, J.: Answer sets for consistent query answering in inconsistent databases. Theory and Practice of Logic Programming 3(4-5), 393–424 (2003)
2. Arieli, O., Denecker, M., Bruynooghe, M.: Distance semantics for database repair. Ann. Math. Artif. Intell. 50(3-4), 389–415 (2007)
3. Arieli, O., Denecker, M., Nuffelen, B.V., Bruynooghe, M.: Computational methods for database repair by signed formulae. Ann. Math. Artif. Intell. 46(1-2), 4–37 (2006)
4. Baldwin, R.W.: Naming and grouping privileges to simplify security management in large databases. In: Proc. 1990 IEEE Symposium on Research in Security and Privacy, pp. 116–132. IEEE Computer Society Press (1990)
5. Bancilhon, F., Spyratos, N.: Independent components of databases. In: Proceedings of the Seventh International Conference on Very Large Data Bases, pp. 398–408 (1981)
6. Bancilhon, F., Spyratos, N.: Update semantics of relational views. ACM Trans. Database Systems 6, 557–575 (1981)
7. Bentayeb, F., Laurent, D.: Inversion de l'algèbre relationnelle et mises à jour. Technical Report 97-9, Université d'Orléans, LIFO (1997)
8. Bentayeb, F., Laurent, D.: View Updates Translations in Relational Databases. In: Quirchmayr, G., Bench-Capon, T.J.M., Schweighofer, E. (eds.) DEXA 1998. LNCS, vol. 1460, pp. 322–331. Springer, Heidelberg (1998)
9. Davey, B.A., Priestly, H.A.: Introduction to Lattices and Order, 2nd edn. Cambridge University Press (2002)

10. Dayal, U., Bernstein, P.A.: On the correct translation of update operations on relational views. ACM Trans. Database Systems 8(3), 381–416 (1982)
11. Hegner, S.J.: An order-based theory of updates for closed database views. Ann. Math. Art. Intell. 40, 63–125 (2004)
12. Hegner, S.J.: Information-based distance measures and the canonical reflection of view updates. Technical Report 0805, Institut für Informatik, Christian-Albrechts-Universität zu Kiel (October 2008); An updated and corrected version, which will appear in Ann. Math. Art. Intell., is available on the Web site of the author
13. Hegner, S.J.: A model of database components and their interconnection based upon communicating views. In: Jakkola, H., Kiyoki, Y., Tokuda, T. (eds.) Information Modelling and Knowledge Systems XIX. Frontiers in Artificial Intelligence and Applications, pp. 79–100. IOS Press (2008)
14. Hegner, S.J.: Semantic Bijectivity and the Uniqueness of Constant-Complement Updates in the Relational Context. In: Schewe, K.-D., Thalheim, B. (eds.) SDKB 2008. LNCS, vol. 4925, pp. 160–179. Springer, Heidelberg (2008)
15. Hegner, S.J.: A Model of Independence and Overlap for Transactions on Database Schemata. In: Catania, B., Ivanović, M., Thalheim, B. (eds.) ADBIS 2010. LNCS, vol. 6295, pp. 204–218. Springer, Heidelberg (2010)
16. Hegner, S.J.: A simple model of negotiation for cooperative updates on database schema components. In: Kiyoki, Y., Tokuda, T., Heimbürger, A., Jaakkola, H., Yoshida, N. (eds.) Frontiers in Artificial Intelligence and Applications XX11, pp. 154–173 (2011)
17. Hegner, S.J., Schmidt, P.: Update Support for Database Views Via Cooperation. In: Ioannidis, Y., Novikov, B., Rachev, B. (eds.) ADBIS 2007. LNCS, vol. 4690, pp. 98–113. Springer, Heidelberg (2007)
18. Keller, A.M.: Updating Relational Databases through Views. PhD thesis, Stanford University (1985)
19. Langerak, R.: View updates in relational databases with an independent scheme. ACM Trans. Database Systems 15(1), 40–66 (1990)
20. Ore, O.: Theory of equivalence relations. Duke Math. J. 9, 573–627 (1942)
21. Osborn, S.L., Guo, Y.: Modeling users in role-based access control. In: ACM Workshop on Role-Based Access Control, pp. 31–37 (2000)
22. Rissanen, J.: Independent components of relations. ACM Trans. Database Systems 2(4), 317–325 (1977)
23. Schewe, K.-D., Thalheim, B.: Component-driven engineering of database applications. In: APCCM 2006: Proceedings of the 3rd Asia-Pacific Conference on Conceptual Modelling, pp. 105–114. Australian Computer Society, Inc., Darlinghurst (2006)
24. Thalheim, B.: Component development and construction for database design. Data Knowl. Eng. 54(1), 77–95 (2005)

SO^F: A Semantic Restriction over Second-Order Logic and Its Polynomial-Time Hierarchy

Alejandro L. Grosso[1] and José M. Turull Torres[2]

[1] Dpto. de Informática, Universidad Nacional de San Luis, San Luis, Argentina
agrosso@unsl.edu.ar

[2] ICTIC, Universidad de la Cuenca del Plata, Corrientes, Argentina
and
Dpto. de Informática, Universidad Nacional de San Luis, Argentina
J.M.Turull@massey.ac.nz

Abstract. We introduce a restriction of second order logic, SO^F, for finite structures. In this restriction the quantifiers range over relations closed by the equivalence relation $\equiv^{FO}$. In this equivalence relation the equivalence classes are formed by k-tuples whose *First Order type* is the same, for some integer $k \geq 1$. This logic is a proper extension of the logic SO$^\omega$ defined by A. Dawar and further studied by F. Ferrarotti and the second author. In the existential fragment of SO^F, $\Sigma_1^{1,F}$, we can express rigidity, which cannot be expressed in SO$^\omega$. We define the complexity class NP^F by using a variation of the relational machine of S. Abiteboul and V. Vianu (RMF) and we prove that this complexity class is captured by $\Sigma_1^{1,F}$. Then we define an RMF^k machine with a *relational* oracle and show the exact correspondence between prenex fragments of SO^F and the levels of the PHF polynomial-time hierarchy.

Keywords: Finite Model Theory, Descriptive Complexity, Relational Machines.

1 Introduction

Significant research has been done in the last few decades in regard to the relationship between finite model theory and computational complexity theory. There is a close relationship between computational complexity, the amount of resources we need to solve a problem over some computation model, and descriptive complexity, the logic we need to describe the problem. The most important result about this relationship was the result of Fagin [4]. This result establishes that the properties of finite structures which are defined by existential second order sentences coincide with the properties that belong to the complexity class NP. This result was extended by Stockmeyer [15] establishing a close relationship between second order logic and the polynomial hierarchy. There are many results equating logic expressibility to computational complexity but they require ordered structures (see [11] [12]) when the target logic is below second order logic. That is, for these logics the correspondence between the expressibility of a logic and the computability with bounded resource does not hold for all classes of finite structures; instead it holds for finite structures which have a linear order as one of its relations. Immerman [10] and Vardi [17] independently demonstrated that the extension of first order logic with a fixed point operator captures PTIME over ordered structures. Such result is not known to hold for arbitrary structures. Similarly, results due to M. Vardi [17] and S. Abiteboul and V. Vianu [1] establish that the extension of first order logic with a partial fixed point operator captures PSPACE over ordered structures.

The interest in fixed point logics attracted attention on the infinitary logic with finitely many variables, $\mathcal{L}^{\omega}_{\infty,\omega}$, since the fixed point logics mentioned are fragments of $\mathcal{L}^{\omega}_{\infty,\omega}$. Definability in this logic can be characterized in terms of a two-player pebble game. (see [11])

In this paper we use a semantic restriction to define the logic SO^F. Similarly to the logic SO^{ω} defined by A. Dawar [2], and further studied by F. Ferrarotti and J. Turull-Torres in [6], where the second order quantifiers range over relations which are closed under the equivalence relation $\equiv^k$, for some k, in SO^F the k-ary second order variables are interpreted by relations closed under the equivalence relation $\equiv^{FO}$ over k-tuples. When we use SO to express a query on a database we extend the database with arbitrary relations for the quantified relation variables. Correspondingly when we use SO^F, we extend the structure with relations closed under FO types. Furthermore when we use SO^{ω}, we extend the structure with relations closed under FO^k types. The quantified relations in SO^F are *redundant* in the sense of [5].

In the equivalence relation $\equiv^k$, two k-tuples are equivalent with respect to a structure $\mathcal{A}$, when they satisfy in $\mathcal{A}$ the same FO^k formulae with k free variables. FO^k is the fragment of FO where the formulae use up to k variables. In the relation $\equiv^{FO}$, two k-tuples are equivalent with respect to a structure $\mathcal{A}$, when they satisfy in $\mathcal{A}$ the same FO formulae with k free variables. If the equivalent classes of $\equiv^{FO}$ have only one k-tuple, the structure is rigid. In this case we can express in SO^F all of SO queries. We have the same situation in SO^{ω} when the structure is FO^k rigid, i.e., when the equivalent classes of $\equiv^k$ have only one k-tuple. Note that if two k-tuples are equivalent in $\equiv^{FO}$ then they are equivalent on $\equiv^k$, but not reciprocally. Y. Gurevich and S. Shela in [9] built rigid structures that are not FO^k rigid for any k. They called this structures *odd multipedes.* Therefore in any class $\mathcal{S}$ of rigid structures, SO^F is equivalent to SO. Similarly, SO^{ω} is equivalent to SO on $\mathcal{F}$ for any class $\mathcal{F}$ of FO^k rigid structures. Then in SO^F, rigidity plays the same role as FO^k rigidity in SO^{ω}.

We show that the Boolean query rigidity is expressible in SO^F, and it is well known that it is not expressible in $\mathcal{L}^{\omega}_{\infty,\omega}$, and hence in SO^{ω}. We prove that every query expressible in SO^{ω} is expressible in SO^F. Then SO^{ω} is strictly included in SO^F. We use a modified relational machine [1] to define the complexity class NP^F. We then prove that this complexity class is captured by the SO^F existential fragment, $\Sigma^{1,F}_1$. The Boolean query rigidity belongs to this fragment.

One of our motivations is to get a better understanding of the distinctive aspects of different NP complete problems. In [8] we prove that some of those problems can be expressed in $\Sigma_1^{1,F}$ (i.e. , that they are in NP^F). We plan to prove that some other problems are not expressible in $\Sigma_1^{1,F}$.

We adapt the relational oracle machine in [6] to the RMF^k oracle machine. Based on this machine we define the PHF polynomial-time hierarchy and show the exact correspondence between the prenex fragments of SO^F and the levels of PHF polynomial-time hierarchy.

This last result is the RMF machine equivalent of the Fagin-Stockmeyer characterization of the polynomial-time hierarchy on Turing machines [15] and also of the Ferrarotti and Turull Torres characterization of the relational polynomial-time hierarchy on relational machines [6].

2 Preliminaries

In our case we consider finite relational structures.

A vocabulary σ is a set of relational symbols $\{P_1, \ldots, P_s\}$ with associated arities $r_1, \ldots, r_s$. A σ-structure (also called model) $\mathcal{A} = \langle A, P_1^{\mathcal{A}}, \ldots, P_s^{\mathcal{A}} \rangle$ consists of a non empty set A called the universe or domain of $\mathcal{A}$ and a relation $P_i^{\mathcal{A}} \subseteq A^{r_i}$ for each relational symbol P_i in σ.

A simple relational structure is a graph $\mathcal{V} = \langle V, E^{\mathcal{V}} \rangle$, where V is the domain and $E^{\mathcal{V}}$ is a binary relation on V, in this case they represent nodes and edges respectively.

A *m-ary query* q is a function from structures of a fixed vocabulary σ to m-ary relations on the domain of the structures, which preserves isomorphisms , i. e., when f is an isomorphism from $\mathcal{A}$ to $\mathcal{B}$ then $\bar{t} \in q(\mathcal{A})$ iff $f(\bar{t}) \in q(\mathcal{B})$. A 0-ary query, also called *Boolean query*, is a function from a class of σ-structures to $\{0,1\}$ and can be identified with a class of σ-structures closed under isomorphisms.

For the definitions of syntax and semantics of FO see [14] among others.

FO^k is the FO fragment where we allow up to k different variables.

In second order (SO) logic we add a set of SO variables, i. e., variables that range over relations on the domain instead of over elements. Then we add for SO the following rules to the FO rules: *If R is a second order variable of arity k, and $x_1, \ldots, x_k$ are first order variables, then $R(x_1, \ldots, x_k)$ is an SO formula (atomic). If φ is a SO formula and R is a second order variable, then $\exists R\varphi$ and $\forall R\varphi$ are SO formulae.* Now we extend the valuation domain to SO variables. A valuation assigns relations of the corresponding arity over the domain to the SO variables. We add the following rules to the FO semantics to obtain the SO semantics. $\mathcal{A} \models R(x_1, \ldots, x_k)[v]$ *iff* $(v(x_1), \ldots, v(x_k)) \in v(R)$. $\mathcal{A} \models \exists R\varphi[v]$ *iff for some* $R^{\mathcal{A}} \subseteq A^k$ *it holds that* $\mathcal{A} \models \varphi[v\frac{R^{\mathcal{A}}}{R}]$, *with k being the arity of R.* $\mathcal{A} \models \forall R\varphi[v]$ *iff for every* $R^{\mathcal{A}} \subseteq A^k$ *it holds that* $\mathcal{A} \models \varphi[v\frac{R^{\mathcal{A}}}{R}]$, *with k being the arity of R and* $v\frac{R^{\mathcal{A}}}{R}$ *es equal to v except* $v\frac{R^{\mathcal{A}}}{R}(R) = R^{\mathcal{A}}$.

2.1 Element Types

For any l-tuple $\bar{a}$ of elements in a σ-structure $\mathcal{A}$, with $1 \leq l \leq k$, we define the FO^k type of $\bar{a}$, denoted $Type_k(\mathcal{A}, \bar{a})$, as the set of FO^k formulae, $\varphi \in FO^k$

with free variables among $x_1, \ldots, x_l$, such that $\mathcal{A} \models \varphi[a_1, \ldots, a_l]$. A set of FO^k formulae τ is an FO^k type, if and only if, it is the FO^k type of some tuple in some structure of the corresponding vocabulary. If τ is an FO^k type, we say that the tuple $\bar{a}$ realizes τ in $\mathcal{A}$, if and only if, $\tau = Type_k(\mathcal{A}, \bar{a})$. We say the type τ is realized in $\mathcal{A}$ when exist a k-tuple $\bar{a}$ in $\mathcal{A}$, such that $\tau = Type_k(\mathcal{A}, \bar{a})$.

Let $\mathcal{A}$, $\mathcal{B}$ be σ-structures and let $\bar{a}$ and $\bar{b}$ be two tuples in the structures $\mathcal{A}$ and $\mathcal{B}$ respectively. $(\mathcal{A}, \bar{a}) \equiv^k (\mathcal{B}, \bar{b})$, if and only if, $Type_k(\mathcal{A}, \bar{a}) = Type_k(\mathcal{B}, \bar{b})$.

For any l-tuple $\bar{a}$ of elements in a σ-structure $\mathcal{A}$, with $1 \leq l \leq k$, we define the FO type of $\bar{a}$, denoted $Type(\mathcal{A}, \bar{a})$, to be the set of FO formulae, $\varphi \in FO$ with free variables among $x_1, \ldots, x_l$, such that $\mathcal{A} \models \varphi[a_1, \ldots, a_l]$.

Let $\mathcal{A}$ and $\mathcal{B}$ be σ-structures and let $\bar{a}$ and $\bar{b}$ be two tuples in the structures $\mathcal{A}$ and $\mathcal{B}$ respectively. We write $(\mathcal{A}, \bar{a}) \equiv^{FO} (\mathcal{B}, \bar{b})$, if and only if, $Type(\mathcal{A}, \bar{a}) = Type(\mathcal{B}, \bar{b})$.

For any structure $\mathcal{A}$ and elements $a_1 \ldots a_l \in dom(\mathcal{A})$, the **basic FO^k type** of $a_1 \ldots a_l$ is the set of FO^k atomic formulae, Φ, with l free variables such that $\mathcal{A} \models \varphi[a_1, \ldots, a_l]$ for every $\varphi \in \Phi$.

Note that for a given finite vocabulary, σ, there are up to equivalence only finitely many distinct basic FO^k types. Furthermore, each basic FO^k type is characterized by a single quantifier free formula of FO^k. The following is a well known fact.

Fact 1. *Let $\mathcal{A}$ be a finite σ-structure and let $\bar{a}$ and $\bar{b}$ be two l-tuplas on $\mathcal{A}$. $(\mathcal{A}, \bar{a}) \equiv^{FO} (\mathcal{A}, \bar{b})$, if and only if, there is an automorphism f such that $f(a_i) = b_i$ for $1 \leq i \leq l$.*

Below, we mention the concept of pre-order and the connection with equivalence relations. We will see that a linear strict pre-order induces an equivalence relation and also establishes a linear order over the equivalence classes.

Definition 1. *Let S be a set, a binary relation R is a **linear strict pre-order** on S if the following holds: 1)$\forall a \in S,\ b \in S,\ aRb \Rightarrow \neg bRa$(asymmetric). 2)$\forall a \in S,\ b \in S,\ c \in S,\ aRb \wedge bRc \Rightarrow aRc$(transitive). 3)$\forall a \in S,\ b \in S,\ c \in S aRb \wedge \neg bRc \wedge \neg cRb \Rightarrow aRc$(linear). In a linear pre-order if the element a is less than an element b, then a is less than every element c non-comparable with b. Note that we can define $a \equiv b$ iff $a \nprec b \wedge b \nprec a$, which is an equivalence relation.*

The following two facts are immediate.

Fact 2. *A **linear strict pre-order** $\prec$ on S induce an equivalence $\equiv$ on S.*

Fact 3. *A **linear strict pre-order** $\prec$ on S induce a total strict order over the equivalence classes of the equivalence relation that such pre-order induces.*

When the equivalence classes of a pre-order on k-tuples from some structure $\mathcal{A}$ agree with the equivalence classes of the equivalence relation $\equiv^k$, then the pre-order establishes a linear order over the FO^k types for k-tuples that are realized on the structure $\mathcal{A}$. We have a similar situation with the relation $\equiv^{FO}$, in this case the pre-order establishes a linear order over the FO types for k-tuples which are realized in $\mathcal{A}$.

Definition 2 (refinement). *Let $\equiv$ be an equivalence relation, $\equiv'$ is a refinement of $\equiv$ iff For all the equivalence classes C_i belonging to $\equiv$, one of the following conditions hold: either C'_i belongs to $\equiv'$ such that $C_i = C'_i$, or there is a partition $C_{i1}, \ldots, C_{ij}$ of C_i such that C_{ih} is an equivalence class that belong to $\equiv'$ for $1 \leq h \leq j$. Furthermore there are no more classes in $\equiv'$, than those described above.*

Definition 3 (agglutination). *Let $\equiv$ be an equivalence relation , $\equiv'$ is an agglutination of $\equiv$ iff $\equiv$ is a refinement of $\equiv'$.*

Let R be a k-ary relation, let $\equiv$ be a $2k$-ary equivalence relation, and let $\bar{x}$ and $\bar{y}$ be k-tuples. R is closed under $\equiv$ iff when $\bar{x} \in R$ and $\bar{x} \equiv \bar{y}$ then $\bar{y} \in R$. That is, the relation R contains complete equivalence classes.

Fact 4. *Let R be a relation closed under $\equiv$, and let $\equiv'$ be a refinement of $\equiv$. Then R is closed under $\equiv'$.*

For the definitions of the fixed point logics LFP, IFP and PFP see [14] among others.

Theorem 1 ([10], [17]). *LFP on ordered structures = P.*

Theorem 2 ([17], [1]). *PFP on ordered structures = PSPACE.*

Definition 4. *$\mathcal{L}_{\infty,\omega}$ is the infinitary extension of first order logic in which we allow conjunctions and disjunctions over arbitrary (not just finite) sets of formulae. $\mathcal{L}^k_{\infty,\omega}$ is the fragment of $\mathcal{L}_{\infty,\omega}$ where we only allow up to k different variables. $\mathcal{L}^\omega_{\infty,\omega} = \bigcup_{k\geq 1} \mathcal{L}^k_{\infty,\omega}$ is the fragment of of $\mathcal{L}_{\infty,\omega}$ where we only use a finite number of variables.*

Kolaitis and Vardi [13] proved that the logics LFP and PFP logics are fragments of $\mathcal{L}^\omega_{\infty,\omega}$. Then, we have:

$$\text{FP} \subseteq \text{PFP} \subset \mathcal{L}^\omega_{\infty,\omega}$$

The last inclusion is strict because in $\mathcal{L}^\omega_{\infty,\omega}$ we can express queries of arbitrary complexity (see [16]).

3 SO^F: A Semantic Restriction of SO

A. Dawar [2] defines a restriction on second order logic by restricting the kind of relations that the quantified second order variables can take. For example we can ask for the relations to be closed under an equivalence relation. Thus A. Dawar [2] defines a restriction of the second order logic, SO$^\omega$, by restricting the second order quantification to relations which are closed under the equivalence relation $\equiv^k$. This imply that we can not quantify arbitrary relations. The relations have to be unions of FO^k types, i.e., they have to be unions of equivalence classes of $\equiv^k$.

Definition 5. *To an l-ary relation symbol R and $k \geq l$, we define the second order quantifier $\exists^k R$ with the following semantic: $\mathcal{A} \models \exists^k R\varphi[v]$ if and only if exists a relation $R^{\mathcal{A}} \subseteq A^l$ such that $R^{\mathcal{A}}$ is closed under the equivalence relation $\equiv^k$, and $\mathcal{A} \models \varphi[v\frac{R^{\mathcal{A}}}{R}]$. As usual $\forall^k R\varphi$ is an abbreviation of $\neg\exists^k R\neg\varphi$. We remember that two l-tuples, of an arbitrary structure $\mathcal{A}$, are equivalent under $\equiv^k$ when they satisfy the same FO^k formulae with l free variables on $\mathcal{A}$.*

Theorem 3 ([2]). $SO^\omega \subseteq PFP$

Therefore SO^w is an effective fragment of $\mathcal{L}^\omega_{\infty,\omega}$.

We define SO^F, a restriction of second order logic, by restricting the second order quantification to relations which are closed under $\equiv^{FO}$, i.e., if the tuple $\bar{a} \in R^{\mathcal{A}}$ and $(\mathcal{A}, \bar{a}) \equiv^{FO} (\mathcal{A}, \bar{b})$ then $\bar{b} \in R^{\mathcal{A}}$. Similarly in our logic we cannot quantify arbitrary relations on a structure $\mathcal{A}$. When we quantify r-ary relations on a structure $\mathcal{A}$ they must be unions of equivalence classes of $\equiv^{FO}$ for r-tuples. Theses relations are redundant in the sense of [5].

Definition 6. *To a k-ary relation symbol R, we define the second order quantifier $\exists^F R$ with the following semantic: $\mathcal{A} \models \exists^F R\varphi[v]$ if and only if exists a relation $R^{\mathcal{A}} \subseteq A^k$ such that $R^{\mathcal{A}}$ is closed under the equivalence relation $\equiv^{FO}$ in $\mathcal{A}$ for k-tuplas, and $\mathcal{A} \models \varphi[v\frac{R^{\mathcal{A}}}{R}]$. As usual $\forall^F R\varphi$ is an abbreviation of $\neg\exists^F R\neg\varphi$.*

We add the following formation rules to FO formation rules to obtain the formulae of SO^F: *1) If R is a k-ary second order variable, and $x_1, \ldots, x_k$ are first order variables, then $R(x_1, \ldots, x_k)$ is a formula (atomic) of SO^F. 2) If φ is a SO^F formula, R is a k-ary second order variable, then $\exists^F R\varphi$ and $\forall^F R\varphi$ are formulae of SO^F.* The fragment $\Sigma^{1,F}_1$ of SO^F consists of the formulae of SO^F which have a prefix of second order existential quantifiers ($\exists^F$) followed by an FO formula.

4 SO^w Is Strictly Included in SO^F

In SO^F we can define an FO type, having in main that an FO type is a minimal relation closed under equivalence relation $\equiv^{FO}$. This minimal relation has empty intersection with the others FO types realized in the structure.

In SO^F we can define an FO type R by saying that every relation S, closed under FO types, includes R or their intersection is empty. $FOt(R) \equiv \forall^F S((R \subseteq S) \vee (R \cap S = \emptyset))$.

Next, we define a linear strict pre-order on k-tuples . In this pre-order the equivalence classes are the FO-types on the structure for k-tuples. We denote as $FOSize_k$ the number of equivalence classes in this pre-order. We denote with $\bar{x}$ a k-tuple $x_1, \ldots, x_k$ of arbitrary variables. With $\mid \bar{x} \mid$ the length of the k-tuple.

Definition 7

$$\begin{aligned}\exists^F \prec^k (\ & \forall\bar{x}\forall\bar{y}\forall\bar{z}\ \ \bar{x} \prec^k \bar{y} \wedge \bar{y} \prec^k \bar{z} \Rightarrow \bar{x} \prec^k \bar{z})\wedge \\ & (\forall\bar{x}\forall\bar{y}\ \ \bar{x} \prec^k \bar{y} \Rightarrow \bar{y} \not\prec^k \bar{x})\wedge \\ & (\forall\bar{x}\forall\bar{y}\forall\bar{z}\ \ \bar{x} \prec^k \bar{y} \wedge \bar{y} \not\prec^k \bar{z} \wedge \bar{z} \not\prec^k \bar{y} \Rightarrow \bar{x} \prec^k \bar{z})\wedge \\ & (\forall\bar{x}\forall\bar{y}\ \ \bar{x} \not\prec^k \bar{y} \wedge \bar{y} \not\prec^k \bar{x} \Rightarrow \exists^F R \\ & (FOt(R) \wedge R(\bar{x}) \wedge R(\bar{y})))\wedge \\ & (\forall^F R\forall\bar{x}\forall\bar{y}(FOt(R) \wedge R(\bar{x}) \wedge R(\bar{y}) \Rightarrow \\ & \bar{x} \not\prec^k \bar{y} \wedge \bar{y} \not\prec^k \bar{x}))\end{aligned}$$

Fact 5 ([7]). *$\prec^k$ is closed under FO types for $2k$-tuples.*

Lemma 1 ([7]). *$LFP_{R,\bar{x}}(\psi(R,\bar{x}))(\bar{t})$ with ψ positive in R is expressible in SO^F.*

Theorem 4 ([3]). *$\equiv^k$ is expressible in LFP with a formula of $2k$ variables.*

Lemma 2 ([7]). *In SO^F we can express the fact $(\mathcal{A},\bar{s}) \equiv^k (\mathcal{A},\bar{t})$.*

The lemma 2 is based on the fact that in SO^F we can express the least fixed point quantifier (LFP) and, for any finite structure $\mathcal{A}$, we can obtain the FO^k types of $\mathcal{A}$ with a formula using LFP quantification.

Theorem 5 ([7]). *$SO^\omega \subseteq SO^F$.*

Corollary 1. *$\Sigma_1^{1,\omega} \subseteq \Sigma_1^{1,F}$.*

Proof. Let $\varphi \equiv \exists R_1^{k-1} \ldots \exists R_n^{k_n} \psi$ be a formula of $\Sigma_1^{1,\omega}$. Then we can express φ in $\Sigma_1^{1,F}$ with the following formula:

$$\begin{aligned}\exists^F R_1 \ldots \exists^F R_n \exists^F \prec^{k_1} \ldots \exists^F \prec^{k_n}\ & \text{“}\prec^{k_1} \textit{ is a pre-order for } k_1\textit{-tuples”} \wedge \\ & \ldots \\ & \text{“}\prec^{k_n} \textit{ is a pre-order for } k_n\textit{-tuples”} \wedge \\ & \forall\bar{x}_1\bar{y}_1(R_1(\bar{x}_1) \wedge \bar{x}_1 \equiv^{k_1} \bar{y}_1 \Rightarrow R_1(\bar{y}_1))\wedge \\ & \ldots \\ & \forall\bar{x}_n\bar{y}_n(R_n(\bar{x}_n) \wedge \bar{x}_n \equiv^{k_n} \bar{y}_n \Rightarrow R_n(\bar{y}_n))\wedge \\ & \psi.\end{aligned}$$

Note that pre-orders can agglutinate the FO types but can not refine the FO types. If the pre-order agglutinate the FO types then the LFP operator must use less stages than $FOSize_k$ to compute the fixed point for the relation $\equiv^k$. □

A structure is rigid if, and only if, the only automorphism that the structure has is the identity function. In a finite rigid structure each element realizes a different FO type. Every k-tuple realizes a different FO type for k-tuples Therefore the cardinality of the equivalence classes of $\equiv^{FO}$ is equal to one. We note that in a finite rigid structure every element can be distinguished from the others by some FO property. Given a rigid finite structure $\mathcal{A}$ every relation in $\mathcal{A}$ is closed under FO types. Thus, SO^F is equivalent to SO on rigid structures.

Lemma 3. *The class of finite rigid stuctures is expressible in $\Sigma_1^{1,F}$.*

Proof. $\exists^F \prec (\prec$ *" is a linear strict pre-order "* $\wedge \forall x \forall y (x \prec y \vee y \prec x \vee x = y)) \Box$

Lemma 4 ([3]). *The class of rigid structures is not definable in $\mathcal{L}^{\omega}_{\infty,\omega}$.*

Since SO$^\omega \subset \mathcal{L}^{\omega}_{\infty,\omega}$ then rigidity is not expressible in SO$^\omega$.

Corollary 2. *$SO^\omega \subset SO^F$.*

5 A Variation of Relational Machine

A relational machine is a Turing machine with a *"relational store"* (rs) where there is a finite number of fixed-arity relations as the input structure. Each transition depends on the symbol which is in the current cell on the input tape, the state at which the machine is, and the result of an FO Boolean query which is evaluated in the rs. If the Boolean query is true the machine eventually goes to a new state, eventually moves the tape head to the right or left and stores the result of an r-ary FO query in rs. A formalization for deterministic relational machines can be found in [6].

We define a modified relational machine to characterize the expressive power of the existential fragment of SO^F, $\Sigma_1^{1,F}$. Let $\mathcal{A}$ be the input structure and S=$\{\prec^{2i} | \prec^{2i}$ *is a linear strict pre-order closed under FO types for $2i$-tuples where their equivalence classes are FO types for i-tuples* , for some $k \geq 1$ and $1 \leq i \leq k\}$. Let $FOSize_k(\mathcal{A})$ *be the number of FO types for k-tuples realized in* $\mathcal{A}$. We define a non-deterministic machine with FO types for k-tuples, RMF^k, as a relational machine with a set of k pre-orders $\{\prec^2, \ldots, \prec^{2k}\} \subseteq S$. The machine begins the computation with the *string* $0^{FOSize_k(\mathcal{A})}1$ in its Turing tape. We note that we have defined in [7] an RMF^k machine version which does not receive the $FOSize_k(\mathcal{A})$ as input in the Turing tape. With that version we can also capture the SO^F existential fragment $\Sigma_1^{1,F}$, but the definition used in this paper is more gently to define the PHF polynomial-time hierarchy of section 6. The following is a formal definition of the machine:

Definition 8. *Let $k \geq 1$. A k-ary non-deterministic relational machine with FO types for k-tuples, denoted as RMF^k, is an 11-tuple $\langle Q, \Sigma, \delta, q_0, \epsilon, F, \tau, \sigma, \upsilon, \Omega, \Gamma \rangle$ where:*

1) Q is a finite set of internal states. 2) Σ is the tape alphabet. 3) $\epsilon \in \Sigma$ is the blank symbol. 4) $q_0 \in Q$ is the initial state. 5) $F \subseteq Q$ is the set of final *states. 6) τ is the rs vocabulary. 7) $\sigma \subset \tau$ is the structure input vocabulary. 8) $\upsilon = \{\prec_1^2, \ldots, \prec_k^{2k}\}$. $\upsilon \subset \tau$ y $\upsilon \cap \sigma = \emptyset$ are k relation symbols instantiated with linear strict pre-orders, such that $\prec_i^{2i}$ has arity $2i$ and the* equivalence classes induced by $\prec_i^{2i}$ are the set of FO types for i-tuples *realized in the input structure $\mathcal{A}$, $1 \leq i \leq k$. 9) Ω is a set of FO sentences of vocabulary τ. 10) Γ is a set of FO formulae of vocabulary τ. 11) $\delta : Q \times \Sigma \times \Omega \to \mathcal{P}(Q \times \Sigma \times \{R, N, L\} \times \tau \times \Gamma)$ is a partial function called the transition function.*

The transitions are based on: i.) the current state; ii.) the content of the current tape cell; and iii.) the answer to a Boolean first-order query evaluated on the τ-structure held in the relational store.

Situations in which the transition function is undefined indicate that the computation must be stop. Otherwise, the result of the transition function is interpreted as follows:

i.) One of the possible 5-tuples of the result set of the function δ is non-determinitically chosen. ii.) the first component is the next state; iii.) the second component is the symbol to be written on the scanned cell of the tape; iv.) the third component specifies the move of the tape head: R means moving one cell to the right, L means moving one cell to the left and N means do not move the head; v.) the fourth component is an r-ary relation symbol of τ, which specifies the r-ary relation to be replaced in the rs with the result of the r-ary query specified in the fifth component, and vi.) the fifth component is an r-ary query in Γ.

We can define *deterministic RMF^k machines* by changing the transition function to: $\delta : Q \times \Sigma \times \Omega \to Q \times \Sigma \times \{R, N, L\} \times \tau \times \Gamma$.

We can introduce for the non-deterministic relational machine with FO types the analogous to the concepts to a configuration and computation of Turing machines.

Let M be an RMF^k, $k \geq 1$. A **configuration** of M is a 3-tuple $\langle q, \omega, I\rangle$, where q is the current internal state of M, $\omega \in \Sigma^* \# \Sigma^*$ represents the current content of the tape and I is the current τ-structure in the rs. The symbol # is suppose not to be in Σ, and marks the position of the tape head (by convention, the head scans the symbol immediately to the right of the #). All cells in the infinite tape not appearing in ω are assumed to contain the particular symbol blank "ϵ".

Let M be an RMF^k, $k \geq 1$. Let $\mathcal{A}$ be a σ-structure. Let $\prec_1^2, \ldots, \prec_k^{2k}$ be k pre-orders on $\mathcal{A}$, where the equivalence classes induced by the preorder $\prec^{2i}$ for $1 \leq i \leq k$ are the FO types for i-tuples realized in $\mathcal{A}$. The **initial configuration** of M with $\mathcal{A}$ as input, is $\langle q_0, \#0^{FOSize_k(\mathcal{A})}1, I\rangle$. Where I is a τ-structure in which $R_i^I = R_i^{\mathcal{A}}$, for $1 \leq i \leq u = \mid \sigma \mid$, $R_{u+i}^I = \prec_i^{2i}$, for $1 \leq i \leq k$ and R_i^I are empty relations i, for $u + k < i \leq l$, $\mid \tau \mid = l$ and $l \geq u + k$.

An **accepting configuration** is a configuration whose state is an accepting state.

Let M a RMF^k, $k \geq 1$, let $\mathcal{A}$ be an input σ-structure. A **partial computation** of M on $\mathcal{A}$ is a (finite or infinite) sequence of configurations of M in which each step from a configuration to the next obeys the transition function.

A **computation** of M is a partial computation which starts with the initial configuration, and ends in a configuration in which no more steps can be performed. An **accepting computation** is a computation ending in an accepting configuration and in this case the input structure $\mathcal{A}$ is **accepted**.

Let M be an RMF^k, $k \geq 1$ and let $\mathcal{A}$ be an input σ-structure, The **computation time** of M for $\mathcal{A}$ is the length of the shortest accepting computation of M. The **computation space** for $\mathcal{A}$ is the minimum number of visited cells in a accepting computation of M.

Definition 9. *Let $\mathcal{L}(M)$ be the relational language accepted by an* RMF^k *machine, M, for $k \geq 1$ and let t and s be functions on the natural numbers such that $t(n) \geq n+1$ and $s(n) \geq 1$. Then, we say that: (i)$\mathcal{L}(M) \in NTIME_F(t(n))$ when the non-deterministic computation time of M for any accepted input structure, $\mathcal{A}$, is bounded above by $t(FOSize_k(\mathcal{A}))$. Where $FOSize_k(\mathcal{A})$ is the number of distinct FO types for k-tuples realized in $\mathcal{A}$. (ii)$\mathcal{L}(M) \in NSPACE_F(s(n))$ when the non-deterministic computation space of M over any accepted input structure, $\mathcal{A}$, is bounded above by $s(FOSize_k(\mathcal{A}))$. Where $FOSize_k(\mathcal{A})$ is as in (i). (iii) Similarly we can define deterministic complexity classes $TIME_F(t(n))$ and $SPACE_F(s(n))$.*

Similarly to the classic complexity class, NP, we can define the complexity class NP^F as the class of the relational languages that are decidable by RMF^k machines that work in non-deterministic polynomial time in the $FOSize_k$ of the input structure. In symbols: $NP^F = \bigcup_{c \in N} NTIME_F(FOSize_k(\mathcal{A})^c)$.

5.1 $\mathbf{NP}^{\boldsymbol{F}} = \boldsymbol{\Sigma}_1^{1,\boldsymbol{F}}$

Theorem 6 ([7]). *Given a $\Sigma_1^{1,F}$ formula $\varphi \equiv \exists^F X_1 \ldots \exists^F X_s \psi(X_1, \ldots, X_s)$, where ψ is a first order formula of vocabulary $\sigma \cup \{X_1, \ldots, X_s\}$ with arities r_j for $1 \leq j \leq s$, we can build an RMF^k $M_\varphi \in NTIME((FOSize_k(\mathcal{A}) \cdot c)$ with $k = max\{r_1, \ldots, r_s\}$ such that for every σ-structure $\mathcal{A}$ we have $\mathcal{A} \models \varphi$ iff $\mathcal{A} \in \mathcal{L}(M_\varphi)$.*

Before proving the converse of the previous result, we need the following fact which is similar to a fact in [6].

Fact 6 ([7]). *Let $\mathcal{A}$ be a relational structure. For $1 \leq i \leq n$, let $k_i \geq 1$ and let $\mathcal{C}_i$ be the set of equivalence classes of k_i-tuples determined by $\equiv^{FO}$ for k_i-tuples on $\mathcal{A}$. It follows that for every $\mathcal{C}_R \subset \mathcal{C}_1 \times \cdots \times \mathcal{C}_n$ the relation*

$$\begin{aligned} R^{\mathcal{A}} = \{(a_{11}, \ldots, a_{1k_1}, \ldots, a_{n1}, \ldots, a_{nk_n}) \in \mathcal{A}^{k_1 + \cdots + k_n} \\ ([a_{11}, \ldots, a_{1k_1}], \ldots, [a_{n1}, \ldots, a_{nk_n}]) \in \mathcal{C}_R\} \end{aligned}$$

is closed under $\equiv^{FO}$ for $(k_1 + \cdots + k_n)$-tuples.

The fact establishes that, if a relation is built by using cartesian product of equivalence classes of $\equiv^{FO}$ for arbitrary arities, then the resulting relation is closed under FO types for the corresponding arity.

In SO^F we can quantify a linear strict pre-order $\prec_k^{2k}$ on k-tuples and we are sure that the relation is closed under FO types for $2k$-tuplas. However, we do not know if the equivalence relation induced by $\prec_k^{2k}$ refines, agglutinates or preserves the FO types for k-tuples. Therefore we need to prove the following lemma:

Lemma 5 ([7]). *Let $\prec$ be a $2k$-ary linear strict pre-order that is closed under FO types for $2k$-tuples. Then, the equivalence relation induced by $\prec$ does not refine the FO types for k-tuples.*

Now we can affirm that if the length of a $2k$-ary linear strict pre-order over a σ-structure $\mathcal{A}$ is equal to $FOSize_k(\mathcal{A})$ then the equivalence classes induced by the pre-order are the FO types for k-tuples realized in $\mathcal{A}$.

We prove the converse of the Theorem 6 by following a similar strategy to [6]:

Theorem 7. *Given an RMF^k $M \in NTIME((FOSize_k(\mathcal{A})) \cdot c)$, for some positive integer c, that computes a Boolean query q, we can build a formula $\varphi_M \in \Sigma_1^{1,F}$ such that M* stops *in an accepting state with $\mathcal{A}$ in the* rs *iff $\mathcal{A} \models \varphi_M$.*

Proof. Let $q : \mathcal{B}_\sigma \longrightarrow \{0,1\}$ be a Boolean query that is computed by the RMF^k machine, $M = \langle Q, \Sigma, \delta, q_0, \epsilon, F, \tau, \sigma, \upsilon, \Omega, \Gamma \rangle$ for some $k \geq 1$, working in linear time in the $FOSize_k$ of the input structure with vocabulary σ. We assume that M works in time $(FOSize_k(\mathcal{A})) \cdot c$ for some $c \geq 1$ and $\mathcal{A} \in \mathcal{B}_\sigma$. Here, 1) $Q = \{q_0, \ldots, q_m\}$. 2) $\Sigma = \{0, 1, \epsilon\}$. 3) $F = \{q_m\}$ 4) $\tau = \{R_0, \ldots, R_l\}$ is the vocabulary of the rs; $r_i (0 \leq i \leq l)$ is the arity of R_i. 5) $\sigma = \{R_0, \ldots, R_{u-1}\}$. 6) $\upsilon = \{R_u, \ldots, R_{u+k-1}\}$ for $u + k \leq l$. 7) $\Omega = \{\omega_0, \ldots, \omega_v\}$ is the finite set of first-order sentences with vocabulary τ. 8) $\Gamma = \{\gamma_0, \ldots, \gamma_t\}$ is the finite set of first-order formulas with vocabulary τ. 9) $\delta : Q \times \Sigma \times \Omega \to \mathcal{P}(Q \times \Sigma \times \{R, N, L\} \times \tau \times \Gamma)$ is the transition function of M. The sentence φ_M expressing acceptance on input $\mathcal{A} \in \mathcal{B}_\sigma$ has the form:

$$\begin{aligned} &\exists^F \prec_1 \ldots \exists^F \prec_k \exists^F T_0 \exists^F T_1 \exists^F T_\epsilon \\ &\quad \exists^F H_{q_0} \ldots \exists^F H_{q_m} \exists^F S_0 \ldots \exists^F S_l (\psi(\prec_1, \ldots, \prec_k T_0, T_1, T_\epsilon, \\ &\qquad\qquad\qquad\qquad\qquad\qquad H_{q_0}, \ldots, H_{q_m}, S_0, \ldots, S_l)). \end{aligned}$$

where $arity(\prec_i) = 2 \cdot i$, for $1 \leq i \leq k$; $arity(T_a) = 2 \cdot \lceil log(c) + 1 \rceil \cdot k$ for $a \in \Sigma$; $arity(H_q) = 2 \cdot \lceil log(c) + 1 \rceil \cdot k$ for $q \in Q$; $arity(S_i) = \lceil log(c) + 1 \rceil \cdot k + arity(R_i)$ for $0 \leq i \leq l$ and ψ is a first-order formula of vocabulary $\{\prec_1, \ldots, \prec_k, T_0, T_1, T_\epsilon, H_{q_0}, \ldots, H_{q_m}, S_0, \ldots, S_l\} \cup \sigma$. The intended interpretation of these relation symbols is as follows:

1) $\prec_i$ is a linear strict pre-order closed under FO types for $2i$-tuples over $\mathcal{A}$. We know that the equivalence classes do not refine the FO types for i-tuples.

Considering that $\prec_k$ is also a linear order over the set, $\mathcal{C}$, of FO types for k-tuples, we can define a lexicographic order over $\lceil log(c) + 1 \rceil$-tuples of $\mathcal{C}^{\lceil log(c)+1 \rceil}$. Since M runs in time bounded by $FOSize_k(\mathcal{A}) \cdot c$ and visits at most $(FOSize_k(\mathcal{A})) \cdot c$ cells, we can model time $(\bar{t})$ as well as position $(\bar{p})$ on the tape by $\lceil log(c)+1\rceil$-tuples of equivalence classes in $\mathcal{C}$. Note that we need find to c' such that $FOSize_k(\mathcal{A})^{c'} = FOSize_k(\mathcal{A}) \cdot c$. Then $c' = 1 + \frac{log(c)}{log(FOSize_k(\mathcal{A}))}$. We assume that $FOSize_k(\mathcal{A}) > 1$ (for $FOSize_k(\mathcal{A}) = 1$ see [7]). Therefore, if we choose $d = \lceil log(c)+1\rceil$, we can code any $\bar{t}$ or $\bar{p}$ with a d-tuple of equivalence classes of $\mathcal{C}$. Under this approach, two $(d \cdot k)$-tuples $\bar{a} = (\bar{a}_1, \dots, \bar{a}_d)$ and $\bar{b} = (\bar{b}_1, \dots, \bar{b}_d)$ are considered equivalent iff, for $1 \leq i \leq d$, $\bar{a}_i \not\prec_k \bar{b}_i \wedge \bar{b}_i \not\prec_k \bar{a}_i$, i.e., iff $\bar{a}_i$ and $\bar{b}_i$ belong to the same equivalence class.

2) T_0, T_1 and T_ϵ are tape relations; for $x \in \Sigma$, $T_x(\bar{p}, \bar{t})$ indicates that the cell $\bar{p}$ at the time $\bar{t}$ contains x.

3) H_{q_i}, $q_i \in Q$, are head relations, $H_{q_i}(\bar{p}, \bar{t})$ indicates that at the time $\bar{t}$ M is in state q_i, and its head is pointing to the cell $\bar{p}$.

4) S_i's are rs relations; for $0 \leq i \leq l$, $S_i(\bar{a}, \bar{t})$ indicates that at time $\bar{t}$ the relation R_i in the rs contains the r_i-tuple $\bar{a}$.

The sentence ψ must now express that when M starts with an input $\mathcal{A}$ in the designed relations in its rs, the $FOSize_k(\mathcal{A})$ in the Turing tape and the relations $\prec_1, \dots, \prec_k$ in the relations $R_u, \dots, R_{u+k-1}$ in its rs, the relations T_x's, H_{q_i}'s and S_i's encode its computation and eventually M reaches an accepting state.

We define ψ to be the conjunction of the following sentences:

1) A sentence expressing that every $\prec_i$ is a linear strict pre-order over i-tuples where by Prop. 5 its equivalence classes do not refine the FO types for i-tuples. $\bigwedge_{1 \leq i \leq k} \prec^i$ is a linear strict pre-order $\wedge$

A sentence defining the initial configuration of M. Let n be the $FOSize_k(\mathcal{A})$. For $\mid \bar{x} \mid = \mid \bar{y} \mid = k \cdot \lceil log(c)+1\rceil$.

$$\exists \bar{x} \forall \bar{y} (\bar{x} \leq \bar{y} \rightarrow (H_{q_0}(\bar{x}, \bar{x}) \wedge \forall \bar{p}(\bar{p} \leq n \rightarrow T_0(\bar{p}, \bar{x})) \wedge \forall \bar{p}(\bar{p} = n+1 \rightarrow T_1(\bar{p}, \bar{x})) \\ \wedge \forall \bar{p}(\bar{p} > n+1 \rightarrow T_\epsilon(\bar{p}, \bar{x})))) \wedge$$

"At time 0, M is in state q_0, the head points to the left-most cell of the tape, and the tape contains $FOSize_k(\mathcal{A})$". Let $\mathcal{C}_0 < \mathcal{C}_1 < \cdots < \mathcal{C}_{FOSize_k(\mathcal{A})-1}$ be the strict order over equivalence classes induced by $\prec^k$. The position n is an d-tuple that belong to $\mathcal{C}_0 \times \mathcal{C}_0 \times \cdots \times \mathcal{C}_0 \times \mathcal{C}_{FOSize_k(\mathcal{A})-1}$. This position can be expressed by using the pre-order $\prec^k$. Similarly for the position $n+1$.

For details of the rest of sentences in the conjunction see [7] □

Lemma 6 ([7]). *Let M be an RMF^k in $NTIME((FOSize_k(\mathcal{A}))^c)$, for some positive integer c, that computes a Boolean query q in polynomial time. Then we can build a formula $\varphi_M \in \Sigma_1^{1,F}$ such that M* stops *in a accepting state with $\mathcal{A}$ in the* rs *iff $\mathcal{A} \models \varphi_M$.*

Corollary 3 ([7]). *Let M be an RMF^k in $NTIME((FOSize_k(\mathcal{A}))^c)$, for some positive integer c, that computes a Boolean query q in polynomial time. Then there exists an $RMF^{k'}$ M' in $NTIME((FOSize_{k'}(\mathcal{A})) \cdot c')$, for $k' > k$, that compute, the Boolean query q in linear time.*

With Corollary 3 we can transform every modified relational machine working in polynomial time in another machine working in linear time. The only consideration is to let the number of FO types grows widening the size of the tuples. Instead of considering FO types for k tuples we consider FO types for k' tuples for some $k' > k$.

6 Oracle Machines and a Polynomial-Time Hierarchy

Definition 10. *Let $k \geq 1$, an RMF^k oracle machine is a RMF^k machine with a distinguished set of relations in its rs, called oracle relations, and three distinguished states $q_?$, the query state, and q_{YES}, q_{NO} the answer states.*

Similarly to the case of oracle Turing machines, the computation of an RMF^k oracle machine requires that an oracle language be fixed previously to the computation. But, since we are working with relational machines, it is natural to think of a *relational oracle language*, i.e., a class of structures of some vocabulary σ^o which is closed under isomorphisms. Let $\mathfrak{S}$ be an arbitrary class of σ^o-structures which is closed under isomorphism. The computation of an RMF^k oracle machine M with oracle $\mathfrak{S}$ and distinguished set of oracle relation symbols σ^o, proceeds like in an ordinary relational machine, except for transitions from the query state. From the query state M transfers into the state q_{YES} if the relational structure of vocabulary σ^o formed by the domain of the input structure and the distinguished set of oracle relations currently held in the rs, belongs to $\mathfrak{S}$; otherwise, M transfers into the state q_{NO}.

Let $k \geq 1$. The time complexity of an RMF^k oracle machine is defined precisely in the same way as with ordinary RMF^k machines. Each query step counts as one ordinary step. Thus if $\mathcal{C}$ is any RMF^k complexity time class, we can define $\mathcal{C}^{\mathfrak{S}}$ to be the class of all relational languages accepted by halting RMF^k machines with oracle $\mathfrak{S}$ that work with time bound as in $\mathcal{C}$.

Definition 11. *Let P^F be $\bigcup_{c \in N} TIME_F(FOSize_k(\mathcal{A})^c)$ for some $k \geq 1$, we define the levels of the PHF polynomial-time hierarchy as follows: 1) $\Delta_0^{P^F} = \Sigma_0^{P^F} = \Pi_0^{P^F} = P^F$ and for $m > 0$*

$$\Delta_{m+1}^{P^F} = P^{F^{\Sigma_m^{P^F}}} \qquad \Sigma_{m+1}^{P^F} = NP^{F^{\Sigma_m^{P^F}}} \qquad \Pi_{m+1}^{P^F} = co-NP^{F^{\Sigma_m^{P^F}}}$$

The PHF complexity class is the union of all relational complexity classes in the PHF polynomial time hierarchy, i.e., PHF$= \bigcup_{m \in \mathbb{N}} \Sigma_m^{P^F}$.

6.1 SO^F Captures the PHF Polynomial-Time Hierarchy

We show the exact correspondence between the prenex fragments of SO^F and the levels of the PHF polynomial-time hierarchy.

Definition 12. *Let $m \geq 1$ and let σ be a relational vocabulary, we denote by $\Sigma_m^{1,F}[\sigma]$ the class of SO^F formulae of the form:*

$$\exists^F X_{11} \ldots \exists^F X_{1s_1} \forall^F X_{21} \ldots \forall^F X_{2s_2} \ldots QX_{m1} \ldots QX_{ms_m}(\varphi)$$

where Q is $\exists^F$ or $\forall^F$, depending on whether m is odd or even, respectively, and φ is an FO formula of vocabulary $\sigma \cup \{X_{11} \dots X_{1s_1} X_{21} \dots X_{2s_2} \dots X_{m1} \dots X_{ms_m}\}$.

Accordingly, we denote by $\Pi^{1,F}_m[\sigma]$ the class of SO^F formulae of the form:

$$\forall^F X_{11} \dots \forall^F X_{1s_1} \exists^F X_{21} \dots \exists^F X_{2s_2} \dots Q X_{m1} \dots Q X_{ms_m}(\varphi)$$

where Q is $\forall^F$ or $\exists^F$, depending on whether m is odd or even, respectively, and φ is an FO formula of vocabulary $\sigma \cup \{X_{11} \dots X_{1s_1} X_{21} \dots X_{2s_2} \dots X_{m1} \dots X_{ms_m}\}$

To prove the following lemma we follow the strategy used in [6].

Lemma 7. *If M is a nondeterministic RMF^k with oracle $\mathfrak{S}$ in $\Sigma^{P^F}_m$ for some $m \geq 0$ and some $k \geq 1$, then there exists a nondeterministic RMF^k machine M' which is equivalent to M and which, in any computation, asks at most one query to an oracle $\mathfrak{S}'$ which is also in $\Sigma^{P^F}_m$.*

Proof. We assume that $M = \langle Q, \Sigma, \delta, q_0, \epsilon, F, \tau, \sigma, \upsilon, \Omega, \Gamma \rangle$ is an RMF^k oracle machine that works in nondeterministic time bounded by $(FOSize_k(\mathcal{A}))^c$ for some $c \geq 1$ and input structure $\mathcal{A}$ of vocabulary $\sigma = \{R_1, \dots, R_n\}$. We also assume that the subset of distinguished oracle relation symbols in the vocabulary τ of M is $\sigma^o = \{E^o_1, \dots, E^o_l\}$ where for $1 \leq i \leq l$ the arity of E_i is r_i and $\sigma \cap \sigma^o = \upsilon \cap \sigma^o = \emptyset$. We denote with $M_{\mathfrak{S}}$ the relational machine in $\Sigma^{P^F}_m$ which decides $\mathfrak{S}$. We assume that $M_{\mathfrak{S}}$ is an RMF^k that works in $(FOSize_k(\mathcal{A}_o))^{c'}$ for some $c' \geq 1$ and input structure $\mathcal{A}_o$ of vocabulary σ^o.

M' works as follows. First, M' guesses a sequence of answers. Let l be the $FOSize_k(\mathcal{A})$. Note that l is written in the Turing tape. M' does this by guessing and writing over its Turing tape a string in $\{Y, N\}^{l^c}$.

Then M' works as M, except that every time that M makes a query to the oracle, M' just takes the answer guessed for the query at the beginning of the computation and adds a set of tuples to oracle relataions. M' has vocabulary $\tau' = \tau \cup \{S^o_1, \dots, S^o_l, \prec^{k^o}\}$ where the subset of distinguished oracle symbol of M' is $\sigma^A = \{S^o_1, \dots, S^o_l, \prec^{k^o}\}$ and for $1 \leq i \leq l$, the arity of S^o_i is $r_i + k \cdot (c+1)$. M' increases the arity of oracle symbols to code the number of query with a $(k \cdot c)$-tuple. Note that the machine M makes as many $(FOSize_k(\mathcal{A}))^c$ queries to the oracle. Let $\mathcal{C}_a$ be the first equivalence class induced by $\prec^k$ when the query answer is no, otherwise $\mathcal{C}_a$ is the last equivalence class. Let $1 \leq m \leq (FOSize_k(\mathcal{A}))^c$, for the m-th query M' adds to the oracle relation S^o_i the following set of $(r_i + k \cdot (c+1))$-tuples: $\{(\bar{e}, \bar{x}) \mid \bar{e} \in E^o_i$ and $\bar{x} = (\bar{x}_1, \dots, \bar{x}_c, \bar{a})$ belongs to $[\bar{x}_1] \times \dots \times [\bar{x}_c] \times \mathcal{C}_a$ where the tuple $([\bar{x}_1], \cdots, [\bar{x}_c])$ is the m-th tuple in the lexicographic order to c-tuples of equivalence classes induced by $\prec^k$ $\}$.

If the sequence of real queries is shorter than the sequence of guessed answers, M' adds the last σ^o-structure computed for M tagged with the query number and the guessed answer to the oracle relations. Note that there is a sequence of guessed answers that repeat until the end the last real answer of M. The oracle relation $\prec^{k^o}$ is initialized with the relation $\prec^k$ of M'.

Note that $M_{\mathfrak{S}'}$ can obtain every σ^o-structure for every query to had to be performed for M. Let $S' \in \mathcal{B}_{\sigma^A}$ and let $\mathcal{C}$ be the set of equivalence classes of the pre-order that interprets the symbol $\prec^{k^o}$. For $([\bar{m}_1], \ldots, [\bar{m}_c], [\bar{m}_{c+1}]) \in \mathcal{C}^{c+1}$ we denote as $Q_{i,([\bar{m}_1],\ldots,[\bar{m}_c],[\bar{m}_{c+1}])}$ the following relation:

$$\{(\bar{a}_1, \ldots, a_{r_i}) \in dom(S')^{r_i} \mid \text{for } \bar{m}'_1 \in [\bar{m}_1], \ldots, \bar{m}'_c \in [\bar{m}_c], \bar{m}'_{c+1} \in [\bar{m}_{c+1}] \text{ it holds that } S' \models S^o_i(\bar{a}_1, \ldots, a_{r_i}, \bar{m}'_1, \ldots, \bar{m}'_c, \bar{m}'_{c+1})\}$$

We denote as $S'_{([\bar{m}_1],\ldots,[\bar{m}_c],[\bar{m}_{c+1}])}$ the σ^o-structure with domain $dom(S')$ which is obtained by interpreting the relation symbol E^o_i with the relation $Q_{i,([\bar{m}_1],\ldots,[\bar{m}_c],[\bar{m}_{c+1}])}$ for $1 \leq i \leq l$.

A given structure $S' \in \mathcal{B}_{\sigma^A}$ belong to the new oracle $\mathfrak{S}'$ if and only if, for every $c+1$-tuple $([\bar{m}_1], \ldots, [\bar{m}_c], [\bar{m}_{c+1}])$ the structure $S'_{([\bar{m}_1],\ldots,[\bar{m}_c],[\bar{m}_{c+1}])} \notin \mathfrak{S}$ and $[\bar{m}_{c+1}]$ is the first equivalence class of the pre-order $\prec^{k^o}$ or $S'_{([\bar{m}_1],\ldots,[\bar{m}_c],[\bar{m}_{c+1}])} \in \mathfrak{S}$ and $[\bar{m}_{c+1}]$ is different of the first equivalence class of the pre-order $\prec^{k^o}$.

At the end of the computation, M' must check that the sequence of guessed answers match the sequence of real answers of M. The RMF^k machine $M_{\mathfrak{S}'}$ which decides the relational language $\mathfrak{S}' \subseteq \mathcal{B}_{\sigma^A}$ works as follows:

1. Let $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_c)$ and $\bar{y} = (\bar{y}_1, \ldots, \bar{y}_c)$ be $(k \cdot c)$-tuples. Let $\bar{a} = (a_1, \ldots, a_k)$ be a k-tuple. Let $\leq^k$ be the lexicographic order induced by $\prec^{k^o}$. We say that $\bar{x} <^k \bar{y}$ when $\bar{x} \leq \bar{y}$ and $\neg \bar{x} \sim \bar{y}$.

```
   $X \leftarrow \forall \bar{y}(\bar{x} \leq^k \bar{y})$
   $Y \leftarrow \neg \bar{x} = \bar{x}$
   **while** $\exists \bar{x}(\neg Y(\bar{x}))$ **do**
      $E_1 \leftarrow \exists \bar{x} \exists \bar{a}(X(\bar{x}) \wedge S^o_1(y_1, \ldots, y_{r_1}, \bar{x}, \bar{a})$
      ...
      $E_l \leftarrow \exists \bar{x} \exists \bar{a}(X(\bar{x}) \wedge S^o_l(y_1, \ldots, y_{r_l}, \bar{x}, \bar{a})$
      $A \leftarrow \exists \bar{x} \exists y_1 \ldots \exists y_{r_l}(X(\bar{x}) \wedge S^o_l(y_1, \ldots, y_{r_l}, \bar{x}, \bar{a})$
      $M_{\mathfrak{S}'}$ works as $M_{\mathfrak{S}}$ taking as input the $\sigma^o$-structure of domain
      $dom(S)$ formed by the relations $R_1, \ldots, R_l$ held in its $rs$
      **if** ($M_{\mathfrak{S}}$ accept $\wedge A$ =first class in $\prec^{k^o}$) $\vee$ ($M_{\mathfrak{S}}$ reject $\wedge A$ $\neq$first class in
      $\prec^{k^o}$) **then**
         $M_{\mathfrak{S}'}$ sotps in a rejecting state
      **end if**
      $Y \leftarrow Y(\bar{x}) \vee X(\bar{x})$
      $X \leftarrow \neg Y(\bar{x}) \wedge (\forall \bar{y}(\bar{y} <^k \bar{x} \rightarrow Y(\bar{y})));$
   **end while**
   $M_{\mathfrak{S}'}$ stops in an accepting state
```

Note that the loop while in the previous algorithm is performed $(FOSize_k(\mathcal{A}))^c$ times because M' builds the oracle structure $\mathcal{S}'$ with unions of FO types realized in the input structure $\mathcal{A}$. Then $\mathcal{S}'$ has the same $FOSize_k$ than $\mathcal{A}$. The same it holds for the oracle structure generated by M. Then is not difficult to see that $M_{\mathfrak{S}'}$ work in nondeterministic time bounded by a polynomial in $FOSize_k(\mathcal{A})$. $\square$

Theorem 8. *For $m \geq 1$, $\Sigma^{1,F}_m$ captures $\Sigma^{P^F}_m$.*

Proof. a)$\Longrightarrow$: We show that for every vocabulary σ, every sentence φ in $\Sigma^{1,F}_m[\sigma]$, φ can be evaluated in $\Sigma^{P^F}_m$.

Suppose that φ is $\exists^F X_{11} \dots \exists^F X_{1s_1} \forall^F X_{21} \dots \forall^F X_{2s_2} \dots\ QX_{m1} \dots QX_{ms_m}$ (ψ), where Q is $\forall^F$ or $\exists^F$, depending on whether m is odd or even, respectively, and ψ is an FO formula of vocabulary $\sigma \cup \{X_{11} \dots X_{1s_1} X_{21} \dots X_{2s_2} \dots X_{m1} \dots X_{ms_m}\}$.

We build a nondeterministic RMF^k oracle machine M_φ which evaluates φ on input structures of vocabulary σ. Let k be $max\{r_{11}, \dots, r_{1s_1}\}$ where r_{1j} is the arity of X_{1j} for $1 \leq j \leq s_1$. The vocabulary τ of the relational store is $\sigma \cup \sigma^{o_{m-1}} \cup \{\prec^1, \dots, \prec^k\} \cup \{S_1, \dots, S_{s_1}\}$ where the arity of S_j is r_{1j} and $\sigma^{o_{m-1}} = \{R^{o_{m-1}} : R \in \sigma\} \cup \{X^{o_{m-1}}_{11}, \dots, X^{o_{m-1}}_{1s_1}\}$ is the set of distinguished oracle symbols. The arity of $R^{o_{m-1}}$ is equal to the arity of R for every $R \in \sigma$, and for $1 \leq j \leq s_1$, the arity of $X^{o_{m-1}}_{1j}$ the same as the arity of X_{1j}.

Let φ_{m-1} be the following sentence: $\exists^F X_{21} \dots \exists^F X_{2s_2} \neg(\exists^F X_{31} \dots \exists^F X_{3s_3} \dots$ $QX_{m1} \dots QX_{ms_m}$ $(\psi_{m-1}))$ where ψ_{m-1} is ψ with every occurrence of a relation symbol $R \in \sigma$ replaced by the corresponding relation symbol $R^{o_{m-1}} \in \sigma^{o_{m-1}}$, and every occurrence of a relation variable X_{1j} $(1 \leq j \leq s_1)$ replaced by the corresponding relation symbol $X^{o_{m-1}}_{1j} \in \sigma^{o_{m-1}}$. The oracle $\mathfrak{S}_{m-1}$ of M_φ is the relational language $\{\mathcal{S} \in \mathcal{B}_{\sigma^{o_{m-1}}} : \mathcal{S} \models \varphi_{m-1}\}$.

On an input structure $\mathcal{A}$, M_φ works as follows:

1) M_φ compute in l_i the length of the pre-order $\prec^i$.

2) For every $1 \leq j \leq s_1$ M_φ guesses and write over its Turing tape a tuple $\bar{a}_j \in \{0,1\}^{l_{r_j}}$ where r_j is the arity of X_{1j}.

3) Using the binary tuples guessed in the previous step, M_φ generates for every $1 \leq j \leq s_1$ a relation of arity r_j which is placed in the distinguished oracle relation $X^{o_{m-1}}_{1j}$ of its relational store rs. M_φ works by storing in $X^{o_{m-1}}_{1j}$ all equivalence classes C_i, where C_i is the ith-equivalence class in the pre-order $\prec^{r_j}$ and the i-th component of $\bar{a}_j$ equals 1. Note that $X^{o_{m-1}}_{1j}$ is closed by FO types for r_j-tuples.

4) Finally, for every $R \in \sigma$, M_φ store the relation $R^{\mathcal{A}}$ into the corresponding oracle relation $R^{o_{m-1}}$ and move to the oracle query state $q_?$. M_φ *accepts* the input structure $\mathcal{A}$ iff the relational structure of domain $dom(\mathcal{A})$ formed by the distinguished oracle relations currently held in its rs does not belong to the oracle set $\mathfrak{S}_{m-1}$, i.e., iff M_φ transfers from the state $q_?$ into the state q_{NO}. If M_φ goes to the state q_{YES} then M_φ *reject.*

M_φ can perform tasks 1 to 3 working in time bounded by a polynomial in $FOSize_k(\mathcal{A})$. Furthermore, task 4 can clearly be performed in constant time by M_φ.

Therefore, it only remain to show that the oracle $\mathfrak{S}_{m-1}$ is in $\Sigma^{P^F}_{m-1}$, i.e, that there is a nondeterministic RMF$^{k'}$ $M_{\varphi_{m-1}}$ such that $\mathcal{L}(M_{\varphi_{m-1}}) = \mathfrak{S}_{m-1}$ and $\mathcal{L}(M_{\varphi_{m-1}}) \in \Sigma^{P^F}_{m-1}$.

$M_{\varphi_{m-1}}$ is an RMF$^{k'}$ oracle machine which evaluates φ_{m-1} on input structures of vocabulary $\sigma^{o_{m-1}}$. Let k' be $max\{r_{21}, \dots, r_{2s_2}\}$ where r_{2j} is the arity of X_{2j}

for $1 \leq j \leq s_2$. The vocabulary τ' of the relational store is $\sigma^{o_{m-1}} \cup \sigma^{o_{m-2}} \cup \{\prec^1, \ldots, \prec^{k'}\} \cup \{S_1, \ldots, S_{s_2}\}$ where the arity of S_j is r_{2j} and $\sigma^{o_{m-2}} = \{R^{o_{m-2}} : R^{o_{m-1}} \in \sigma^{o_{m-1}}\} \cup \{X_{21}^{o_{m-2}}, \ldots, X_{2s_2}^{o_{m-2}}\}$ is the set of distinguished oracle symbols. The arity of $R^{o_{m-2}}$ is equal to the arity of $R^{o_{m-1}}$ for every $R^{o_{m-1}} \in \sigma^{o_{m-1}}$, and for $1 \leq j \leq s_1$, the arity of $X_{2j}^{o_{m-2}}$ the same as the arity of X_{2j}.

Let φ_{m-2} be the following sentence: $\exists^F X_{31} \ldots \exists^F X_{3s_3} \ldots QX_{m1} \ldots QX_{ms_m}$ $(\psi_{m-2}))$ where ψ_{m-2} is ψ_{m-1} with every occurrence of a relation symbol $R \in \sigma^{o_{m-1}}$ replaced by the corresponding relation symbol $R^{o_{m-2}} \in \sigma^{o_{m-2}}$, and every occurrence of a relation variable X_{2j} $(1 \leq j \leq s_2)$ replaced by the corresponding relation symbol $X_{2j}^{o_{m-2}} \in \sigma^{o_{m-2}}$. The oracle $\mathfrak{S}_{m-2}$ of $M_{\varphi_{m-1}}$ is the relational language $\{\mathcal{S} \in \mathcal{B}_{\sigma^{o_{m-2}}} : \mathcal{S} \models \varphi_{m-2}\}$.

The way in which the machine $M_{\varphi_{m-1}}$ works is exactly the same as the way in which the machine M_φ works. $M_{\varphi_{m-1}}$ perform steps 1 to 4 adapted to the vocabulary τ' of its rs and for $1 \leq j \leq s_2$. Therefore, for every input structure $\mathcal{A}'$ of vocabulary $\sigma^{o_{m-1}}$, $M_{\varphi_{m-1}}$ works in nondeterministic time bounded by a polynomial in $FOSize_{k'}(\mathcal{A}')$.

This process continues in the same way for the blocks 3 to $m-1$ of quantifiers in φ. Since for the last block m of quantifiers the resulting φ_1 is either $\exists^F X_{1m} \ldots \exists^F X_{ms_m}(\psi_1)$ or $\exists^F X_{1m} \ldots \exists^F X_{ms_m}(\neg\psi_1)$.

it follows by Theorem 6 that the oracle $\mathfrak{S}_1$ of M_{φ_2} (i.e., the relational language $\{\mathcal{S} \in \mathcal{B}_{\sigma^{o_1}} : \mathcal{S} \models \varphi_1\}$) is in $\mathrm{NP}^{F^{\Sigma_0^{P^F}}}$.

Hence φ can be evaluated in $\Sigma_m^{P^F}$.

b)$\Longleftarrow$: Next, we show that every $\Sigma_m^{P^F}$ property of finite relational structures can be expressed in $\Sigma_m^{1,F}$.

We use induction on m. The base case is Theorem 7. Now consider a Boolean query $q : \mathcal{B}_\sigma \longrightarrow \{0, 1\}$ in $\Sigma_m^{P^F}$ where $m > 1$. Let M be the nondeterministic RMF^k machine with an oracle in $\Sigma_{m-1}^{P^F}$ which computes q. Let $\{S \in \mathcal{B}_{\sigma^o} : q_o(S) = 1\}$, where σ^o is the set of distinguised oracle relation symbols of M and q_o is a Boolean query in $\Sigma_{m-1}^{P^F}$, be the oracle of M.

By inductive hypothesis, for every Boolean query q_i in $\Sigma_{m-1}^{P^F}$, there is a sentence α_{q_i} in $\Sigma_{m-1}^{1,F}$ which expresses q_i. In particular there is a sentence α_{q_o} in $\Sigma_{m-1}^{1,F}$, of vocabulary σ^o, which expresses q_o. Let α_{q_o} be the σ^o-sentence $\exists^F X_{21} \ldots \exists^F X_{2s_2} \forall^F X_{31} \ldots \forall^F X_{3s_3} \ldots QX_{m1} \ldots QX_{ms_m}(\psi_o)$, where Q is either $\exists^F$ or $\forall^F$, depending on m is odd or even, respectively, and ψ_o is a first-order sentence of vocabulary $\sigma^o \cup \{X_{21}, \ldots, X_{2s_2}, X_{31}, \ldots, X_{3s_3}, \ldots, X_{m1}, \ldots, X_{ms_m}\}$.

We show how to modify the formula φ_M of Theorem 7, i.e., the formula corresponding to the nondeterministic RMF^k machine without oracle, to reflect the interaction of M with its oracle. We assume that M works in time $(FOSize_k(\mathcal{A}))^c$ for some $c \geq 1$ and $\mathcal{A} \in \mathcal{B}_\sigma$.

First, we add to the prefix of φ_M the existential quantification $\exists^F S_1^o \ldots \exists^F S_n^o$. Let r_i^o denote the arity of the distinguished oracle relation symbol R_i^o in $\sigma^o = \{R_1^o, \ldots, R_n^o\}$. For $1 \leq i \leq n$, the arity of the relation variable S_i^o is $r_i^o + k \cdot c$. The intended interpretation of $S_i^o(\bar{x}, \bar{t})$ is that at time $\bar{t}$ the distinguished oracle relation R_i^o in the rs contains the r_i^o-tuple $\bar{x}$.

The sub-formula ψ of φ_M treats the variables $S^o_1, \ldots, S^o_n$ corresponding to the distinguished oracle relation in the rs of M in exactly the same way as the variable $S_1, \ldots, S_l$ which correspond to the other relation in the rs of M. We only need to add a special case to the sub-formula ψ which express that the relations T_i's, H_q's, S_i's and S^o_i's obey the transition function of M. When M is in the oracle query state $q_?$, $\chi(q, a, \alpha, q', c, m, R, \gamma)$ is the sentence discribing the transition in which upon entering the query state $q_?$, the machine moves to state q_{YES} if the σ^o-structure held in the rs is in the oracle of M, or to state q_{NO} if it is not. W.l.o.g we assume that the contents of the rs as well as of the Turing tape of M and the position of its read/write head remain unchanged.

The more "natural" way of expressing $\chi(q, a, \alpha, q', c, m, R, \gamma)$ is probably as follows:

$$\begin{aligned}\forall \bar{p} \forall \bar{t} (H_{q_?}(\bar{p}, \bar{x}) \to & (\hat{\alpha}_{q_o}(\bar{t}) \to H_{q_{YES}}(\bar{p}, \bar{t}+1)) \wedge (\neg \hat{\alpha}_{q_o}(\bar{t}) \to H_{q_{NO}}(\bar{p}, \bar{t}+1)) \wedge \\ & \forall \bar{p}' (\bigwedge_{i \in \{0,1,\epsilon\}} T_i(\bar{p}', \bar{t}+1) \leftrightarrow T_i(\bar{p}', \bar{t})) \wedge \\ & \bigwedge_{1 \leq i \leq l} (\forall \bar{x} (S_i(\bar{x}, \bar{t}+1) \leftrightarrow S_i(\bar{x}, \bar{t}))) \wedge \\ & \bigwedge_{1 \leq i \leq n} (\forall \bar{x} (S^o_i(\bar{x}, \bar{t}+1) \leftrightarrow S^o_i(\bar{x}, \bar{t}))))\end{aligned}$$

where $\hat{\alpha}_{q_o}$ is the formula obtained by replacing in α_{q_o} each atomic sub-formula of the form $R^o_i(x_1, \ldots, x_{r^o_i})$ by $S^o_i(x_1, \ldots, x_{r^o_i}, \bar{t})$ for $1 \leq i \leq n$. But we need the resulting formula to be in prenex normal form.

In order to write $\chi(q, a, \alpha, q', c, m, R, \gamma)$ in a form such that the SO^F quantifiers in α_{q_o} can be moved to the prefix of φ_M, we use Lemma 7. That is, we assume that in any computation M makes at most one query to its oracle. Under this assumption, we can then write $\chi(q, a, \alpha, q', c, m, R, \gamma)$ as the conjunction of: $\exists^F X_{21} \ldots \exists^F X_{2s_2} \forall^F X_{31} \ldots \forall^F X_{3s_3} \ldots Q X_{m1} \ldots Q X_{ms_m}$

$$\begin{aligned}\forall \bar{p} \forall \bar{t} (H_{q_?}(\bar{p}, \bar{x}) \wedge \hat{\psi}_o(\bar{t}) \to & H_{q_{YES}}(\bar{p}, \bar{t}+1) \wedge \\ & \forall \bar{p}' (\bigwedge_{i \in \{0,1,\epsilon\}} T_i(\bar{p}', \bar{t}+1) \leftrightarrow T_i(\bar{p}', \bar{t})) \wedge \\ & \bigwedge_{1 \leq i \leq l} (\forall \bar{x} (S_i(\bar{x}, \bar{t}+1) \leftrightarrow S_i(\bar{x}, \bar{t}))) \wedge \\ & \bigwedge_{1 \leq i \leq n} (\forall \bar{x} (S^o_i(\bar{x}, \bar{t}+1) \leftrightarrow S^o_i(\bar{x}, \bar{t}))))\end{aligned}$$

and

$$\forall^F X'_{21} \ldots \forall^F X'_{2s_2} \exists^F X'_{31} \ldots \exists^F X'_{3s_3} \ldots QX'_{m1} \ldots QX'_{ms_m}$$

$$\begin{aligned}\forall \bar{p} \forall \bar{t} (H_{q_?}(\bar{p}, \bar{x}) \wedge \neg \hat{\psi}'_o(\bar{t}) \rightarrow & H_{q_{NO}}(\bar{p}, \bar{t}+1) \wedge \\ & \forall \bar{p}' (\bigwedge_{i \in \{0,1,\epsilon\}} T_i(\bar{p}', \bar{t}+1) \leftrightarrow T_i(\bar{p}', \bar{t})) \wedge \\ & \bigwedge_{1 \leq i \leq l} (\forall \bar{x} (S_i(\bar{x}, \bar{t}+1) \leftrightarrow S_i(\bar{x}, \bar{t}))) \wedge \\ & \bigwedge_{1 \leq i \leq n} (\forall \bar{x} (S^o_i(\bar{x}, \bar{t}+1) \leftrightarrow S^o_i(\bar{x}, \bar{t}))))\end{aligned}$$

where $\hat{\psi}_o(\bar{t})$ is the formula obtained by replacing in ψ_o each atomic sub-formula $R^o_i(x_1, \ldots, x_{r^o_i})$ $(1 \leq i \leq n)$ by $S^o_i(x_1, \ldots, x_{r^o_i}, \bar{t})$ and $\hat{\psi}'_o$ is the formula obtained by replacing in $\hat{\psi}_o$ each occurrence of a relation variable $X_{ij} \in \{X_{21}, \ldots, X_{2s_2}, X_{31}, \ldots, X_{3s_3}, \ldots, X_{m1}, \ldots, X_{ms_m}\}$ by X'_{ij}. Note that for the sentence above, we use the fact that $\neg \alpha_{q_o}$ is equivalent to

$$\forall^F X_{21} \ldots \forall^F X_{2s_2} \exists^F X_{31} \ldots \exists^F X_{3s_3} \ldots QX_{m1} \ldots QX_{ms_m} (\neg \psi_o)$$

It is not difficult to see that the SO^F quantifiers in the sentence above can now be safely moved to the prefix of φ_M and rearranged in such a way that the resulting formula is in $\Sigma^{1,F}_m$. □

7 Conclusion

We have defined a new restricted second order logic in which we can express a significative query like rigidity. We have defined a modified relational machine to characterize the existential fragment of this logic. We must emphasize that $NTIME((FOSize_k(\mathcal{A}))^c) = NTIME((FOSize_{k'}(\mathcal{A})) \cdot c')$ for some $k' > k$, so that to have polynomial time (beyond linear time) is of little utility in RMF machines. It is known that rigidity belongs to co-NP, but it is not known if it belongs to NP. Since rigidity belongs to NP^F then it seems that NP$^F \not\subseteq$ NP.

On this new machine we define an oracle machine on which basis we define a hierarchy PHF, similar to polynomial hierarchy, PH, and the relational polynomial hierarchy, PHr [6]. With this framework we are defining a new notion of complexity, analogous to Turing machine complexity and relational complexity. The complexity classes are characterized by the RMF^k relational machine with oracles.

Furthermore, we plan to define a non-deterministic fixed point quantifier, NFPF, in order to establish a result linking non-deterministic fixed points with increasing levels of negations with the PHF hierarchy, in an analogous way to the development made by A. Dawar in [2] for SO$^\omega$ and NFP (non-deterministic fixed point).

References

1. Abiteboul, S., Vianu, V.: Datalog extensions for database queries and updates. J. Comput. System Sci. 43, 62–124 (1991)
2. Dawar, A.: A Restricted Second Order Logic for Finite Structures. Information and Computation 143, 154–174 (1998)
3. Dawar, A.: Feasible Computation through Model Theory. Ph.D. thesis, University of Pennsylvania (1993)
4. Fagin, R.: Generalized first-order spectra and polynomial-time recognizable sets. Complexity of Computation 7, 43–73 (1974); Karp, R.M. (ed.) SIAM-AMS Proceedings
5. Ferrarotti, F.A., Paoletti, A.L., Turull Torres, J.M.: Redundant Relations in Relational Databases: A Model Theoretic Perspective. The Journal of Universal Computer Science 16(20), 2934–2955 (2010)
6. Ferrarotti, F.A., Turull Torres, J.M.: The Relational Polynomial-Time Hierarchy and Second-Order Logic. In: Schewe, K.-D., Thalheim, B. (eds.) SDKB 2008. LNCS, vol. 4925, pp. 48–76. Springer, Heidelberg (2008)
7. Grosso, A.L., Turull Torres, J.M.: A Second-Order Logic in which Variables Range over Relations with Complete First-Order Types. In: 2010 XXIX International Conference of the Chilean Computer Science Society (SCCC), pp. 270–279. IEEE (2010)
8. Grosso, A.L., Turull Torres, J.M.: A Survey on Semantic Restrictions of Second Order Logic (2011) (in preparation)
9. Gurevich, Y., Shela, S.: On finite rigid structures. Journal of Symbolic Logic 61 (1996)
10. Immerman, N.: Relational queries computable en polynomial time. Inform. and Control 68, 86–104 (1986)
11. Immerman, N.: Descriptive and computational complexity. In: Hartmanis, J. (ed.) Proc. of AMS Symposia in Appl. Math. Computational Complexity Theory, vol. 38, pp. 75–91 (1989)
12. Immerman, N.: Descriptive Complexity. Springer, Heidelberg (1998) ISBN 0-387-98600-6
13. Kolaitis, P., Vardi, M.: Infinitary logics and 0-1 laws. Information and Commputation 98(2), 258–294 (1992)
14. Libkin, L.: Elements of Finite Model Theory. Springer, Heidelberg (2004) ISBN 3-5402-1202-7
15. Stockmeyer, L.: The polynomial-time hierarchy. Theoret. Comput. Sci. 3, 1–22 (1976)
16. Turull-Torres, J.M.: A study of homogeneity in relational databases. Ann. Math. Artif. Intell. 33(2-4), 379–414 (2001), See also Erratum for: A Study of Homogeneity in Relational Databases. Annals of Mathematics and Artificial Intelligence 42, 443-444 (2004)
17. Vardi, M.Y.: The complexity of relational query languages. In: Proc. 14th ACM Symposium on the Theory of Computing, pp. 137–146 (1982)

Abstract State Machines for Data-Parallel Computing

Qing Wang

University of Otago, Dunedin, New Zealand
qing.wang@otago.ac.nz

Abstract. The current programming paradigm for data-parallel computations is shifting with the rapidly increasing data growth on the web. It gives programmers more challenges than ever before. In this paper we propose Parallel Abstract State Machines (P-ASMs) that can empower programmers, no matter how experienced, by providing a well-founded systems engineering method to model data-parallel computations at arbitrary levels of abstraction. Particularly, we focus on discussing how P-ASMs can capture two classes of data-parallel computations that are most important in practice – ones that are always-consistent and ones that require transactional data consistency.

1 Introduction

In recent years, the "NoSQL" community emerges, targeting to solve problems that can be better handled in distributed and non-relational databases rather than using traditional SQL technologies. A good number of NoSQL database solutions have since been developed, for example, document-oriented database MongoDB [15], distributed storage systems HBase [13], Amazon SimpleDB [3] and Google Bigtable [9], etc. On the other hand, success stories of the MapReduce model [11] in large-scale parallel data-processing (e.g. Google and Facebook) has also opened up debate about current limitations of traditional database management systems (DBMSs) in data-parallel computing. In a nutshell, the challenges faced by traditional DBMSs are how to efficiently and effectively handle data-processing tasks over large data sets on the web.

With the MapReduce model being considered as a more complementary than competing technology to DBMSs [20], integrating the MapReduce model with DBMSs under a hybrid framework that can retain the advantages of both sides has attracted much research effort [1,8,16,21,24]. One of central themes in this area of research is dealing with declarative languages for data-parallel computations. For example, programming languages like Pig Latin [16] and HIVE [21] provide a set of declarative constructs (e.g., filtering, grouping, joining, etc.) following the spirit of SQL, which are executed over an open-source, map-reduce implementation. Conversely, Aster Data [4] extends the expressiveness of SQL (e.g., click stream sessionization) by adding a suite of SQL-MapReduce functions within an integrated database environment. Despite that these data-parallel programming languages vary considerably in design choices, they commonly specify data transformations at a high but fixed level of abstraction. This hinders, to some extent, the flexibility of programming parallel computations because programmers have no freedom to choose the granularity of programming based on their own expertise.

We present Parallel Abstract State Machines (P-ASMs) in this paper, which capture the programming paradigm for data-parallel computations at flexible levels of abstraction. This is due to the power inherited from Abstract State Machines (ASMs), i.e., a universal computation model that can formalise the notion of algorithm [7,12,22,19]. P-ASMs provide the programming capability for both programmers who lack experience in parallel computing and programmers who have more expertise. Starting from specifying high-level data-processing tasks, programmers can stepwise refine their tasks until reaching certain level of abstraction with which they are satisfied, and then leave the rest of implementation details to the underlying system. This process of stepwise refinement makes it possible to verify properties of data-processing tasks at a high-level even after the data-processing tasks have been refined to the implementation level.

Many data-parallel programming languages concern about read-only data analytic tasks such as querying over crawled documents and web log records. Nevertheless, there are some data-processing tasks requiring the guarantee of transaction data consistency to certain degree. Extending previous results from distributed database transactions into a web computing environment often has to trade off some transactional properties or performance for scalability. For example, transactional systems CloudTPS [23] and ElasTraS [10] support scalable transactions for web applications in the cloud by using a collection of transaction managers responsible for disjoint partitions of data sets. However, globally ordering the execution of conflicting transactions across all transaction managers in CloudTPS [23] degrades the performance as scale goes up, and the transactional semantics of ElasTraS [10] is limited by the design of dynamically partitioning a data set at the schema level. The systems Sinfonia [2] and Percolator [18] enable transactional access to (in-memory or persistent) data across a distributed system based on a message passing mechanism. Sinfonia [2] provides a low-level data access interface to support transactions with weak semantics, while Percolator [18] explicitly maintains locks causing significant overhead.

P-ASMs provide a lightweight approach for data-parallel transactions satisfying the transaction properties ACID (i.e., atomic, consistent, isolated and durable) at the snapshot isolation level [5]. Compared with those approaches, P-ASMs have several advantages. First, unlike Sinfonia [2], P-ASMs can specify data-parallel transactions at arbitrary level of abstraction (i.e, high-level, mid-level and low-level). Second, conflicting transactions are dynamically clustered in P-ASMs by the transaction engine and assigned to transaction managers based on the analysis of key pairs of transactions, so it is different from CloudTPS [23] (i.e., using timestamps to order conflicting transactions) and ElasTraS [10] (i.e., managing dynamically partitions at the schema level). Third, P-ASMs do not need to explicitly maintain locks as Percolator [18] because conflicting transactions are managed by the same transaction manager.

The rest of the paper is structured as follows. In Section 2 we discuss several major properties that are often taken into consideration when determining the underlying storage systems for data-processing tasks. Then the formal definitions of P-ASMs are introduced in Section 3. After that, Section 4 and Section 5 present how to use P-ASMs for specifying two classes of data-parallel computations – always-consistent computations and computations satisfying certain level of transaction data consistency. The paper is concluded in Section 6.

2 Storage Choices

A first important step in designing data-parallel programs is to decide how data should be persistently stored since the format of data will put severe limits on the search of optimal solutions. This requires the analysis of not only the data structure of application domains but also the targeted properties of data-processing tasks (e.g., performance, scalability, fault tolerance, data consistency, etc.) that are often tied to architectural differences among data-parallel computing systems such as DBMSs, NoSQL databases, the MapReduce model, etc. The following are several common questions that are taken into consideration when searching for an appropriate storage strategy for data-processing tasks.

How much semantics is necessary? Semantics, in principle, would help improve data quality and performance. However, it does not come free. Before leveraging powerful semantic technologies, data must be properly parsed and loaded. This shifts some of the computational cost at run time to loading time. Thus, varying from different kinds of data-processing tasks, an enough-but-not-excessive level of semantic granularity needs to be decided on data storage models. Traditional DBMSs offer a strong capability base for handling data semantic constraints, file systems mainly store raw data, and NoSQL databases lie in between, capturing data semantics in a way specific to certain kinds of data-processing tasks.

Which level of data consistency is needed? Traditional DBMSs can guarantee strong data consistency, which normalise data to avoid update anomalies and minimise conflicts between concurrent transactions. By contrast, most of NoSQL database solutions de-normalise data, providing a lightweight model for handling consistency - generally not providing immediately consistent or multi-object transactions (e.g., MongoDB and Bigtable). Thus, again, depending on data consistency requirements arising from data-processing tasks, a tradeoff between the level of data consistency and performance has to be made. This would affect how data are persisted into a storage system, e.g., the degree of normalization.

To what extent is the performance scalable? Supported by various optimisation techniques, traditional DBMSs often have superior performance for a large class of data-processing tasks [17]. Nevertheless, it is still a big challenge for DBMSs to scale well to large data sets because of rapidly increasing complexity of control after the number of computing nodes goes up to some extent. It is unknown whether a general DBMS solution at web scale is feasible. In general, NoSQL databases and the MapReduce model achieve high scalability for specific kinds of data-processing tasks meanwhile sacrificing some less important functionality.

3 Parallel Abstract State Machines

In this section we present the formal definitions of Parallel Abstract State Machines (P-ASMs).

Definition 1. *A* parallel abstract state machine $\Lambda = (\mathcal{S}, S_0, S_f, r, \delta)$ *over signature* Υ *consists of a non-empty set* $\mathcal{S}$ *of states over* Υ *together with an initial state* $S_0 \in \mathcal{S}$ *and a final state* $S_f \in \mathcal{S}$*, a rule* r *over* Υ *and a one-step transformation* δ *over* $\mathcal{S}$*, i.e.,* $\delta : \mathcal{S} \mapsto \mathcal{S}$*.*

3.1 States

Following ASM's standard definitions, each state of a P-ASM can be viewed as a first-order structure. A *signature* Υ is a set of function symbols, each associated with a fixed arity. A *state* over Υ consists of a set B, called the *base set* of the state, together with interpretations of all function symbols in Υ. A function symbol in Υ is *dynamic* if the function is changeable in states; otherwise, it is *static*. A pair $\langle f, (a_1, ..., a_n)\rangle$ is called a *location* for the dynamic function symbol f and the tuple $(a_1, ..., a_n)$ of elements in the base set. The value $f(a_1, ..., a_n)$ over state S is the *content* of the location in S.

This abstract point of view on states gives us the freedom to capture various data structures that may co-exist within a state of P-ASMs, e.g., in-memory data structures, persistent data structures in file and database systems, etc. Without loss of generality, we consider that states of a P-ASM are associated with certain type system. A type may be regarded as a collection of values having a uniform structure. This enables us to describe complex values (e.g. list, map, array) in data structures in terms of the type constructors provided by the chosen type system.

Example 1. A simple type system can be defined as

$$\tau = \tau_\lambda \mid \tau_{\mathbf{b}} \mid (A_1 : \tau_1, \ldots, A_k : \tau_k) \mid \{\tau\} \mid [\tau] \mid [\![\tau]\!] \mid \tau_1 \sqcup \tau_2$$

where τ_λ is a trivial type denoting the empty set $\emptyset$, $\tau_{\mathbf{b}}$ represents a base type such as INT, STRING, DATE, etc., and $(\cdot)$, $\{\cdot\}$, $[\cdot]$, $[\![\tau]\!]$ and $\sqcup$ are the type constructors for record with arity k, finite set, finite list, finite multiset and binary union, respectively. □

3.2 Updates

An *update* of Λ is a pair (ℓ, b), where ℓ is a location and b is called the *update value* of ℓ. An *update set* Δ is a set of updates. An update set Δ is *consistent* if it does not contain conflicting updates, i.e. for all $(\ell, b), (\ell, b') \in \Delta$ we have $b = b'$. If S is a state of the P-ASM Λ and Δ is a consistent update set for the signature of Λ, then there exists a unique state $S' = S + \Delta$ resulting from updating S with Δ:

$$val_{S+\Delta}(\ell) \quad = \quad \begin{cases} b & \text{if } (\ell, b) \in \Delta \\ val_S(\ell) & \text{else} \end{cases}$$

In addition to standard updates as described before, P-ASMs also allow *partial updates* in the form of (ℓ, b, ρ), where ℓ is a location, b is called the *partial update value* of ℓ and ρ is an *update operator*. The update operator ρ can be considered as an aggregate function (α, β) defined by $\rho(m) = \alpha(b_1) \odot \cdots \odot \alpha(b_n)$ for $m = \{\!\{b_1, ..., b_n\}\!\}$ is a multiset of elements $b_1, \ldots, b_n$, α is a unary function and β is a commutative and associative binary operation. To distinguish from partial updates, we call standard updates as *total updates*. An update multiset M is a multiset of total and partial updates, which can be reduced to an update set Δ_M containing only total updates such that

$$\Delta_M = \{(\ell, b) | b = \rho(\biguplus_{(\ell, b', \rho) \in M} \{\!\{b'\}\!\})\} \cup \{(\ell, b) | (\ell, b) \in M\}.$$

The sequential composition of two update sets Δ_1 and Δ_2 is an update set defined as: $\Delta_1 \oslash \Delta_2 = \Delta_2 \cup \{(\ell, b) \in \Delta_1 | \neg \exists b'.(\ell, b') \in \Delta_2\}$.

3.3 Rules

A set of P-ASM rules over signature Υ can be inductively defined in the following.

– *update rule:* The content of the location $f(t_1, \ldots, t_n)$ over Υ is updated to the value t_0.
$$f(t_1, \ldots, t_n) := t_0$$
– *conditional rule:* If φ is true, then execute the rule r over Υ; otherwise do nothing.
$$\textbf{if } \varphi \textbf{ then } r \textbf{ endif}$$
– *block rule:* The rules r_1,...,r_n over Υ are executed in parallel.
$$\textbf{par } r_1 \ldots r_n \textbf{ endpar}$$
– *forall rule:* The rule r over Υ is executed in parallel for each x satisfying φ over Υ.
$$\textbf{forall } x \textbf{ with } \varphi \textbf{ do } r \textbf{ enddo}$$
– *partial update rule:* The content of location $f(t_1, \ldots, t_n)$ over Υ is partially updated by the value t_0 and the operator ρ.
$$f(t_1, \ldots, t_n) :=^{\rho} t_0$$
– *call rule:* The rule r over Υ is called with the parameters $t_1, \ldots, t_n$ over Υ.
$$r(t_1, \ldots, t_n)$$

The sequential ASM thesis by Gurevich [12] has proven that every sequential algorithm can be captured by a sequential ASM being comprised of the update, conditional and block rules. The sequential rule $r_1; \ldots; r_n$ appeared in most of programming languages has been excluded from sequential ASMs for the reason of keeping a clear abstraction level. Nevertheless, the sequential rule can be easily incorporated into sequential ASMs and thereby P-ASMs as a syntactic sugar. The forall and partial update rules play an important role in P-ASMs because they can formalise a number of parallel computations bound to the underlying state (i.e., data structures) and aggregate partial updates yielded from these parallel computations. The introduce of the call rule is to provide a flexible abstraction of rules in P-ASMs.

For a P-ASM $\varLambda = (\mathcal{S}, S_0, S_f, r, \delta)$, the one-step transformation δ is determined by the *closed* rule r, i.e., each variable in r is bounded by a forall rule. Each closed rule r executing over a state S is associated with an update multiset and an update set $\varDelta(r, S)$ reduced from the update multiset. There are two types of semantics for P-ASMs: *standard* and *transactional* semantics. The standard semantics of P-ASMs requires that the one-step transformation δ is determined by applying all updates yielded by r over the state S_i such that $\delta(S_i) = S_i + \varDelta(r, S_i)$. Under the transactional semantics, the one-step transformation δ is determined by applying (possibly a subset of) updates yielded by r over the state S_i that must comply with the transactional properties such that $\delta(S_i) - S_i \subseteq \varDelta(r, S_i)$ and $\delta(S_i) - S_i = \varDelta_1 \oslash \cdots \oslash \varDelta_n$ where $\varDelta_i \subseteq \varDelta(r, S_i)(i = 1, ...n)$ is a set of updates yielded by a transaction executed in r over S_i, and $\varDelta_1 \dots \varDelta_n$ are yielded by different transactions.

Definition 2. *A* run *of a P-ASM $\varLambda = (\mathcal{S}, S_0, S_f, r, \delta)$ is a finite sequence $S_0, \dots, S_n$ of states with $S_n = S_f$, S_i and $S_{i+1} \in \mathcal{S}$, and $\delta(S_i) = S_{i+1}$ for all $i = 0, \dots, n-1$.*

4 Always-Consistency

We now explore how to capture a class of data-processing tasks that are always-consistent by using P-ASMs.

Formally, a P-ASM $\varLambda = (\mathcal{S}, S_0, S_f, r, \delta)$ is said to be *always-consistent* iff, for every state $S \in \mathcal{S}$, the update set $\varDelta(r, S)$ yielded by r over S is consistent. The motivations of discovering always-consistent P-ASMs are twofold. First, always-consistent P-ASMs capture data-parallel computations arising from a large body of applications in practice, e.g., OLAP queries, web analytics, indexing, sorting, etc. Second, always-consistent P-ASMs can be easily parallelised at an appropriate level of granularity.

In order to efficiently decide if or not a P-ASM is always-consistent, we introduce several inference rules in Fig. 1. Given an update u, the notations $\sigma(u)$ and $\upsilon(u)$ denote the location and the (partial) update value of u, respectively. If u is a total update, then $\theta(u)$ is undefined; if u is a partial update (ℓ, b, ρ), then $\theta(u) = \rho$. $\varDelta(r)$ denotes the set of (total or partial) updates yielded by r at any state, and $r[x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n]$ denotes a rule after substituting the variables $x_1, \dots, x_n$ occurring in the rule r with the parameters $t_1, \dots, t_n$, correspondingly.

As shown in Fig. 1, inconsistencies can only arise from parallel rules - either forall rules or block rules. The equation (4) states that a block rule is always-consistent if any two updates on the same location yielded by rules $r_1, \dots, r_n$ over any state either have the same update value, or have the same operator (in the later case, they must be partial updates). Similarly, the equation (5) states that a forall rule is always-consistent if any two updates on the same location yielded by rules $r[x \leftarrow t_i](i = 1, \dots, n)$ for each t_i satisfying φ over any state have either the same update value or the same operator.

$$\frac{}{f(t_1, \dots, t_n) := t_0} \tag{1}$$

$$\frac{}{f(t_1, \dots, t_n) :=^{\rho} t_0} \tag{2}$$

$$\frac{r}{\textbf{if } \varphi \textbf{ then } r} \tag{3}$$

$$\frac{r_1, \dots, r_n}{\textbf{par } r_1 \dots r_n \textbf{ endpar}} \quad \begin{array}{l} \forall u_1, u_2 \in \bigcup\limits_{0 \le i \le n} \Delta(r_i).(\sigma(u_1) = \sigma(u_2) \\ \Rightarrow (\upsilon(u_1) = \upsilon(u_2) \vee \theta(u_1) = \theta(u_2))) \end{array} \tag{4}$$

$$\frac{r[x \leftarrow t_1], \dots, r[x \leftarrow t_n]}{\textbf{forall } x \textbf{ with } \varphi \textbf{ do } r \textbf{ enddo}} \quad \begin{array}{l} \forall u_1, u_2 \in \bigcup\limits_{0 \le i \le n} \Delta(r[x \leftarrow t_i]).(\sigma(u_1) = \sigma(u_2) \\ \Rightarrow (\upsilon(u_1) = \upsilon(u_2) \vee \theta(u_1) = \theta(u_2))) \\ \qquad \text{where } \{\!\{x|\varphi(x)\}\!\} = \{\!\{t_1, \dots, t_n\}\!\} \end{array} \tag{5}$$

$$\frac{r'[x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n]}{r(t_1, \dots, t_n)} \quad r(x_1, \dots, x_n) = r' \text{ is a rule declaration} \tag{6}$$

Fig. 1. Inference rules for always-consistency

Parallelising always-consistent P-ASMs is important for improving the performance of data-processing tasks running in a large-scale environment. However, the kinds of parallelisms imposed by the block and forall rules are different. A block rule **par** $r_1 \dots r_n$ **endpar** enables control parallelism in which the rules $r_1, \dots, r_n$ can be distributed across parallel computing nodes, and the degree of parallelism is bounded. A forall rule **forall** x **with** φ **do** r **enddo** allows data parallelism that executes the same rule r on different chunks of the data $\{x|\varphi\}$ across parallel computing nodes, and the degree of parallelism is unbounded, i.e., depending on the underlying state. When a computing node becomes idle, the chunks assigned to computing nodes that are slowly processing or have failures are split again and re-assigned to the idle computing node. This split-assign process is recursively applied to achieve load balancing among a cluster of parallel computing nodes.

Total and partial updates are also handled differently in the process of parallelising always-consistent P-ASMs. Total updates from parallelised subcomputations are materialised into the underlying state as early as possible to utilise the maximum computing resources. However, partial updates should not be persistent until they are combined into total updates at the end of each one-step transformation. The reason for this is to avoid a race condition for locations of partial updates and thus to reduce I/O contention. Using hash partitioning based on locations, we can combine partial updates by distributing them over different computing nodes. If the number of distinct locations involved is relatively small, partial updates may first be locally combined at each computing node before being combined with ones from remote computing nodes. In doing so, we can reduce the network traffic between parallel computing nodes.

P-ASMs can apply the MapReduce model [11] to automatically generate map and reduce jobs over a cluster of computing machines. There are two functions used in the MapReduce model: a map function taking a pair of key and value to generate a set of key and value pairs, and a reduce function merging all values having the same key to produce a possibly smaller set of values. The following example illustrates the differences between a native implementation and a MapReduce implementation specified by P-ASMs.

Example 2. Let us consider the query of page ranks over the table $urls = \{url, category, pagerank\}$, taken from [16]:

select *category*, avg(*pagerank*) from *urls* where *pagerank* > 0.2
group by *category* having count(*)$> 10^6$

Fig. 2 (a) describes the query that would use a native implementation of P-ASMs and Fig. 2 (b) defines a pair of map and reduce functions for the same query. Two approaches suit for solving different problems. If the relation *urls* has only a small number of categories and the records in *urls* that have the same category are potentially large, then the query in Fig. 2 (a) would be more efficient because partial updates can be easily combined at each local computing machine. By contrast, partially merging the key and value pairs $(category, pagerank)$ in Fig. 2 (b) is more complicated since the records having the same category also need be counted. Certain form of sequentiality is used in both approaches. The query in Fig. 2 (a) consists of two sequential steps: calculating f_{sum} and f_{count} for each category, and then calculating f_{avg} for the selected categories, while in Fig. 2 (b) the pairs of key and value generated from the map function need to be grouped by the keys before moving into the reduce function. Nevertheless, both approaches need to handle the problem of “straggler” [11] (i.e., computing nodes whose performance is extremely poor in comparison with others). □

```
forall $category$, $pagerank$, $url$ with $urls(url, category, pagerank) \wedge pagerank > 0.2$ do
  par
    $f_{sum}(category) :=^{+} pagerank$
    $f_{count}(category) :=^{+} 1$
  endpar
enddo;
forall $category$ with $f_{count}(category) > 10^6$ do
  $f_{avg}(category) := f_{sum}(category)/f_{count}(category)$
enddo
```

(a)

```
map($key$, $urls$) $\equiv$
forall $category$, $pagerank$, $url$ with $urls(url, category, pagerank) \wedge pagerank > 0.2$ do
  $f(new) := (category, pagerank)$
enddo

reduce($key$, list($value$)) $\equiv$
// $key$: a category
// list($value$): a list of pageranks for that category
forall $pagerank$ with $pagerank \in$list$(value)$ do
  par
    $f_{sum}(category) :=^{+} pagerank$
    $f_{count}(category) :=^{+} 1$
  endpar
enddo;
if $f_{count}(category) > 10^6$ then
  $f_{avg}(category) := f_{sum}(category)/f_{count}(category)$
endif
```

(b)

Fig. 2. An example of page ranks

5 Transactional Data Consistency

Applications involving updates on data persisted in a storage system often require certain level of transactional data consistency. A typical scenario is an accounting application, in which an amount of money is transferred from the account of a payer to the account of a payee, as will be illustrated in Fig. 3.

P-ASMs provide a lightweight approach to handle a class of transactions with the following properties. 1) Transactions are simple in the sense that only a limited number of operations involved in a transaction. Consequentially, the update set yielded by such a transaction is small. 2) Parallel transactions are generated on the basis of manipulating different chunks of the data, so called *data-parallel transactions*. 3) Transactions comply with the ACID properties at the snapshot isolation level [5]. 4) Transactions are not always-consistent. It means that these transactions may conflict each other in various ways, depending on the underlying state.

To cope with the atomicity property (i.e., the update set yielded by each transaction of P-ASMs must be either all applied or none of them are applied on a state), the transactional semantics of P-ASMs relaxes the standard semantics of P-ASMs to allow the persistence for some of updates in one-step transformations when the desired ACID properties are preserved in each transaction. The following transaction rule is used for specifying the rule r executed as a transaction (i.e., under transactional semantics) in P-ASMs, where (rkey, wkey) is a pair of read and write keys. A (read or write) key can be composite, consisting of a set of key values.

begin(rkey, wkey) r **end**

Example 3. The P-ASM in Fig. 3 iteratively generates data-parallel transactions to handle incoming bank transactions in the table $tra = \{payer, payee, amount\}$. The updates on the account balances of a payer and a payee in an accounting transaction are included within one transaction. The read key is *payer* because the account balance of a payer needs to be checked, which involves the read operation. Since the account balances of both a payer and a payee need to be updated, the write key consists of *payer* and *payee*. There is no need to consider the changes on the tables *log* (i.e., adding a log record) and *tra* (i.e., removing a finished accounting transaction) in the key pair because these changes do not cause any conflicts among transactions. □

Thus, each transaction t of P-ASMs is associated with a key pair ($rkey(t)$, $wkey(t)$) indicating the read and write data items that may cause conflicts, respectively. The intention of introducing a key pair to each transaction is to facilitate the transaction engine to efficiently detect potential conflicts of data-parallel transactions. Based on detected conflicts, the transaction engine assigns transactions to a number of transaction managers. The principles are as follows: a) parallelising transactions by clustering all potentially conflicting transactions together meanwhile distributing transaction clusters as evenly as possible among transaction managers, and b) detecting actual conflicts among transactions as early as possible so that the transactions conflicting with transactions that have already committed can abort or retry again.

Compared with the optimistic concurrency transaction control [14], our approach extends it in two ways. First, using the key pairs to ensure a serialised execution of conflicting transactions and resolve conflicts before materialising data can help reduce I/O contention on storage systems. Second, separating transactions into clusters without any conflicts among each other can improve the transaction throughout since only conflicting transactions are serialised.

In the following, we sketch a general picture of how transactions are handled in P-ASMs. The transaction engine of P-ASMs is responsible for coordinating all transaction-related activities. At the beginning of a transaction, it sends the key pair to the transaction engine, and receives a unique transaction ID and snapshot timestamp that are respectively generated from the ID and timestamp services of the transaction engine. While transactions are executed by a set of computing nodes in parallel, the transaction engine classifies them into different clusters by

```
if $mode = running$ then
  forall $payer, payee, amount$ with $tra(payer, payee, amount)$ do
    begin$(payer, (payer, payee))$
      if $acc(payer) \geq amount$ then
        par
          $acc(payer) := acc(payer) - amount$
          $acc(payee) := acc(payee) + amount$
          $log(new) := (payer, payee, amount, time)$
          $tra(payer, payee, amount) := false$
        endpar
      endif
    end
  enddo
endif
```

Fig. 3. A simple example of accounting transactions

analysing their key pairs. From a graphic point of view, a transaction cluster can be represented by a *cluster graph* (T, γ) consisting of a set T of transactions and a set γ of edges between transactions such that all transactions in T are interconnected in the cluster graph. Furthermore, the sets of transactions from any two different transaction clusters are disjoint, i.e., $\bigwedge\limits_{1 \leq i \neq j \leq n} T_i \cap T_j = \emptyset$ for the set $\{(T_1, \gamma_1), \ldots, (T_n, \gamma_n)\}$ of transaction clusters associated with a P-ASM executed under the transactional semantics.

As transactions of P-ASMs are considered at the snapshot isolation level, a *test* predicate that detects the read-write and write-write conflicts between two transactions $t^{'}$ and t can be defined as: $test(t^{'}, t) \equiv (rkey(t^{'}) \cap wkey(t) \neq \emptyset) \vee (wkey(t^{'}) \cap wkey(t) \neq \emptyset) \vee (rkey(t) \cap wkey(t^{'}) \neq \emptyset)$. The following is the pseudo code of an algorithm used to cluster a set of transactions. We use $C = \{(T_i, \gamma|T_i)|T_i \subseteq T^{'}\}$ to denote the set of transaction clusters where $\bigcup\limits_{1 \leq i \leq n} T_i = T^{'}$ and $(T_i, \gamma|T_i)$ is a subgraph of $(T^{'}, \gamma)$ on the vertex set T_i.

Input: A set of transactions $T = \{t_1, \ldots, t_m\}$
Output: A set of transaction clusters $C = \{(T_i, \gamma|T_i)|T_i \subseteq T^{'}\}$

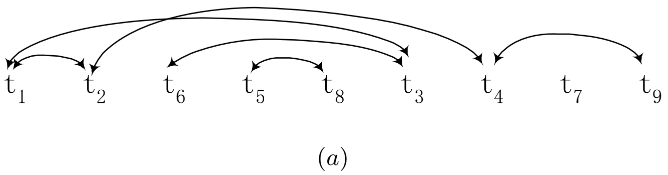
```
par
  if mode=init then
    par
      $\gamma := \emptyset$
      $T^{'} := \{\{t\}|t \in T\}$
    endpar;
    forall $t_1, t_2$ with $\{t_1, t_2\} \subseteq T \wedge test(t_1, t_2)$ do
      $\gamma^{'} :=^{\cup} \{\{t_1, t_2\}\}$
    enddo;
    par
```

```
            $\gamma := \gamma \cup \gamma^{'}$
            mode:=iterate
        endpar
    endif
    if mode=iterate then
        forall $p$ with $p \in \gamma$ do
            $p^{'} := \bigcup\{x|x \in T^{'}$ and $x \cap p \neq \emptyset\}$;
            if $p^{'} \in T^{'}$ then
                mode:=finish
            endif;
            $T^{'} := T^{'} \cup \{p^{'}\} - \{x|x \subseteq p^{'}$ and $x \in T^{'}\}$
        enddo
    endif
endpar
```

Transaction clusters are dynamically managed by the transaction engine via two kinds of operations: *merge* and *split*. Let C be a set of transaction clusters. If an incoming transaction conflicts at least one transaction in each cluster of a subset $C^{'} \subseteq C$ of transaction clusters, then all transactions in $C^{'}$ waiting for commit are merged into one cluster. The split operation often happens when a transaction has been accomplished (either committed or aborted). The accomplished transaction may lead to other transactions in the same cluster split into two clusters between which no transactions are conflicting each other.

The transaction engine schedules the execution of transaction clusters across a set of transaction managers by taking into account the availability of system resources, the locality of data items and the requirements of fault tolerance, etc. A transaction manager uses the two-phase commit protocol (2PC) [6] to manage the commit/abort process of transactions. If the commit of a transaction succeeds, other active transactions conflicting with the committed one have to abort. If the transaction fails to commit and aborts, the transaction manager starts to handle another transaction from the assigned cluster. Since all updates pertaining to the potentially conflicting transactions would be managed by the same transaction manager, the communication cost among different transaction managers is minimised.

Example 4. Consider the example of accounting transactions described in Fig. 3. For simplicity, we assume that there are 9 accounting transactions in the table *tra* and consequently 9 data-parallel transactions $\{t_i|i = 1, \ldots, 9\}$ are executed. We also assume that each of the transactions receives a snapshot timestamp in an order as shown in Fig. 4(a), in which the bidirectional arrows indicate the potential conflicts between the transactions. Fig. 4 (b) illustrates how the transactional engine clusters transactions at three particular points in time (from left to right). At the first point there are three clusters $\{\{t_1, t_2\}, \{t_6\}, \{t_5\}\}$ where t_1 and t_2 are conflicting transactions. During the step (i), all the transactions after t_5 as shown in Fig. 4(a) are added into different clusters. We assume that none of 9 transactions have started to commit at this time. Thus two clusters $\{t_1, t_2\}$ and $\{t_6\}$ are merged into one with other transactions t_3, t_4 and t_9 since t_3 conflicts with both t_1 and t_6 at the second point. Assume that all transaction managers have the same speed to handle transactions. Then, during the step (ii), the transactions t_1, t_7 and t_5 commit. Then the transactions t_2 and t_3 have to abort due to the conflicts with t_1. In order to improve performance, the transactions t_6, t_4 and t_9 in a cluster can be split into two clusters $\{t_6\}$ and $\{t_4, t_9\}$ managed by two different transaction managers, as described at the third point in Fig. 4(b).



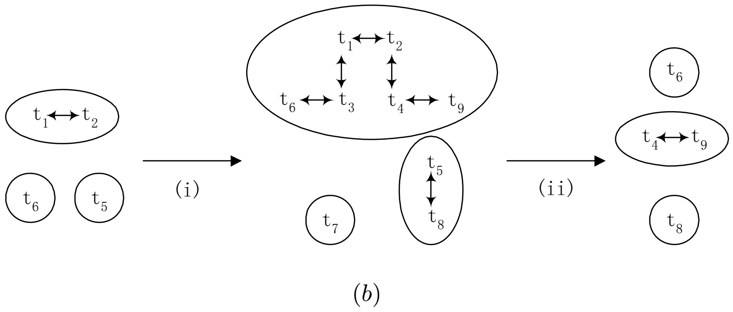


Fig. 4. Sample of data-parallel transactions

6 Conclusion

A model of computation called P-ASMs has been developed in this paper, which can capture data-parallel computations at flexible levels of abstraction. We have discussed how P-ASMs handle always-consistent computations and data-parallel transactions that satisfy the ACID transactional properties at the snapshot isolation level. In the future, we intend to implement the execution and transaction engine of P-ASMs, and to evaluate how efficiently P-ASMs can help solve problems in the data-parallel computing paradigm.

References

1. Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D., Silberschatz, A., Rasin, A.: HadoopDB: An architectural hybrid of MapReduce and DBMS technologies for analytical workloads. Proceedings of the VLDB Endowment 2(1), 922–933 (2009)
2. Aguilera, M., Merchant, A., Shah, M., Veitch, A., Karamanolis, C.: Sinfonia: a new paradigm for building scalable distributed systems. In: Proceedings of Twenty-First ACM SIGOPS Symposium on Operating Systems Principles, pp. 159–174. ACM (2007)
3. Amazon.com. Amazon SimpleDB (2010), http://aws.amazon.com/simpledb
4. Aster Data, http://www.asterdata.com/mapreduce/
5. Berenson, H., Bernstein, P., Gray, J., Melton, J., O'Neil, E., O'Neil, P.: A critique of ansi sql isolation levels. SIGMOD Rec. 24, 1–10 (1995)
6. Bernstein, P.A., Hadzilacos, V., Goodman, N.: Concurrency control and recovery in database systems, vol. 5. Addison-Wesley, New York (1987)
7. Börger, E., Stärk, R.F.: Abstract State Machines: A Method for High-Level System Design and Analysis. Springer-Verlag New York, Inc., Heidelberg (2003)
8. Chaiken, R., Jenkins, B., Larson, P.Å., Ramsey, B., Shakib, D., Weaver, S., Zhou, J.: Scope: easy and efficient parallel processing of massive data sets. Proceedings of the VLDB Endowment 1(2), 1265–1276 (2008)
9. Chang, F., Dean, J., Ghemawat, S., Hsieh, W., Wallach, D., Burrows, M., Chandra, T., Fikes, A., Gruber, R.: Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS) 26(2), 1–26 (2008)
10. Das, S., Agrawal, D., Abbadi, A.E.: Elastras: An elastic transactional data store in the cloud. In: USENIX HotCloud. USENIX (June 2009)
11. Dean, J., Ghemawat, S.: MapReduce: Simplified data processing on large clusters. Communications of the ACM 51(1), 107–113 (2008)
12. Gurevich, Y.: Sequential abstract-state machines capture sequential algorithms. ACM Trans. Comput. Log. 1(1), 77–111 (2000)
13. HBase: Bigtable-like structured storage for Hadoop HDFS (2009), http://hadoop.apache.org/hbase/
14. Kung, H., Robinson, J.: On optimistic methods for concurrency control. ACM Transactions on Database Systems (TODS) 6(2), 213–226 (1981)
15. MongoDB, http://mongodb.org
16. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig Latin: a not-so-foreign language for data processing. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1099–1110. ACM (2008)
17. Pavlo, A., Paulson, E., Rasin, A., Abadi, D.J., DeWitt, D.J., Madden, S., Stonebraker, M.: A comparison of approaches to large-scale data analysis. In: Proceedings of the 35th SIGMOD International Conference on Management of Data, SIGMOD 2009, pp. 165–178. ACM (2009)
18. Peng, D., Dabek, F.: Large-scale incremental processing using distributed transactions and notifications. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, OSDI 2010, pp. 1–15. USENIX Association (2010)
19. Schewe, K.-D., Wang, Q.: A customised ASM thesis for database transformations. Acta Cybern. 19, 765–805 (2010)
20. Stonebraker, M., Abadi, D., DeWitt, D., Madden, S., Paulson, E., Pavlo, A., Rasin, A.: MapReduce and parallel DBMSs: friends or foes? Communications of the ACM 53(1), 64–71 (2010)

21. Thusoo, A., Sarma, J., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., Murthy, R.: Hive: a warehousing solution over a map-reduce framework. Proceedings of the VLDB Endowment 2(2), 1626–1629 (2009)
22. Wang, Q.: Logical Foundations of Database Transformations for Complex-Value Databases. Logos Verlag, Berlin (2010)
23. Wei, Z., Pierre, G., Chi, C.: CloudTPS: Scalable transactions for Web applications in the cloud (2010)
24. Yu, Y., Isard, M., Fetterly, D., Budiu, M., Erlingsson, Ú., Gunda, P., Currey, J.: DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language. In: Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation, pp. 1–14. USENIX Association (2008)

OSM-Logic: A Fact-Oriented, Time-Dependent Formalization of Object-oriented Systems Modeling

Stephen W. Clyde[1], David W. Embley[2], Stephen W. Liddle[3], and Scott N. Woodfield[2]

[1] Computer Science Department
Utah State University, Logan, Utah 84322, USA
[2] Department of Computer Science
[3] Information Systems Department,
Brigham Young University, Provo, Utah 84602, USA

Abstract. The lack of fact-oriented, time-dependent formalizations of conceptual models leads to difficulties in inspecting and reasoning about system properties and predicting future behavior from past behavior. We can better serve these needs by formalized conceptualizations that more closely match the demands of such applications. We therefore set forth in this chapter a fact-oriented, time-dependent formalism, called *OSM-Logic*, for object existence, object interrelationships, object behavior, and object interaction. OSM-Logic is formally grounded in predicate calculus, and is thus mathematically sound and well defined.

1 Introduction

Recent initiatives by government agencies (e.g., IARPA [IAR]) and by academic think-tank groups (e.g., ACM-L [ACM]) require conceptualizations with the ability to track behavior, model what has already happened and is currently happening, and analyze past and present behavior. The objectives of tracking, modeling, and analyzing include being able to predict future behavior, play out "what-if" scenarios, and warn of possible impending disasters.

Conceptual models can provide the formal foundation for storing the necessary information to support these initiatives. The conceptual models that meet these requirements, however, must be powerful: they must be able to conceptualize objects, relationships among objects, object behavior, and object interaction, and the conceptualizations must be fact-oriented and time-dependent. Without being able to formalize and store time-dependent facts about objects—their interrelationships, their individual behavior, and their interaction with other objects—analysis of, and predictions based on, current, past, and proposed happenings cannot be carried out. We thus seek for, and propose here, a formalization of fact-oriented, time-dependent conceptualizations of objects—their existence, their interrelationships, their behavior, and their interactions with other objects.

To define precisely what we mean by *fact-oriented, time-dependent conceptualizations*, we note that conceptual modelers have observed that fact-oriented modeling focuses on facts of interest that can be expressed as first-order-logic predicates [ORM]. Further, for every fact, in addition to knowing *if* it is true, we should know *when* it is true. Since facts hold at points in time or over a period of time, we obtain time-dependent facts by adding to every logic predicate, one argument for a time variable for facts that hold for a point in time and two time-variable arguments for facts that hold over a period of time. Thus, as the basis for our formalism, we seek for first-order-logic predicates augmented with either one time variable for point-in-time facts or two time variables for time-period facts. Further, all conceptualizations—including objects, relationships among objects, object behavior, and object interactions—should be formalized with fact-oriented, time-dependent first-order-logic predicates.

A number of conceptual-model formalizations have been developed as evidenced by hundreds of articles and books ranging from the earliest abstract formulations from more than 50 years ago [YK58] through books that encapsulate much of the 50-year history of conceptual modeling (e.g., [Oli07]) to a recent handbook of conceptual modeling taking formal foundations for granted and as being expected [ET11]. Although plentiful and useful for their intended applications, none of the formalizations have all the characteristics required for fact-oriented, time-dependent conceptualizations of object and relationship existence, object behavior, and object interaction for the applications we target in this chapter. Only a few conceptual models even span the space from object existence through object behavior and interaction, and of those that do, even fewer have formal definitions. Those that span the space and have worked-out or mostly worked-out formalisms include the Unified Modeling Language (UML) [UML], Object Process Modeling (OPM) [Dor09], Object Role Modeling (ORM) [HM08], the software production environment for MDA based on the OO-Method and OASIS [PM07], the Higher-order Entity Relationship Model (HERM) [Tha00], and Object-oriented Systems Modeling (OSM) [EKW92]. Even though these conceptual models have formalizations that span the space from object existence to object behavior, most of the behavior formalizations are neither fact-oriented nor time-dependent, but are, instead, based on ideas from state charts, finite state machines, and Petri nets.

The OSM (and ORM) formalisms are predicate-calculus based and thus fact-oriented, but neither intrinsically has time-dependent conceptualizations. Although predicate calculus does not inherently involve time, with some work we can extend predicate calculus to a two-sorted first-order logic that captures notions of time, including events, time intervals, and time dependencies. Indeed, this is the contribution of this chapter, where we formally define fact-oriented, time-dependent semantics for OSM by showing how to convert any OSM model instance to formulas in *OSM-Logic* [Cly93] and how to interpret these formulas.

Briefly and succinctly, OSM-Logic is a two-sorted, first-order logic language with temporal semantics specifically developed for defining the meaning of OSM model instances. Since OSM captures static and dynamic properties of real-world systems, OSM-Logic must be able to express relationships among objects, object behavior, and interactions with respect to time. In real-world systems, time involves continuous intervals, (t_1, t_2) and individual points, (t). Most changes in a system of objects occur during time intervals and are not instantaneous. Even simple changes like an object entering a new state or becoming a member of an object set occur over a time interval. In fact, for any interpretation of a system, a change that appears instantaneous may actually be occurring during a time interval that is simply smaller than smallest unit of time in that interpretation. By using a finer unit of time, what once appeared instantaneous can appear as a time interval. On the other hand, an event is a single time point that represents a true instantaneous occurrence. An event typically corresponds to the beginning or end of a particular time interval during which something interesting occurs.

Compared to the vast amount of work on temporal models, the static model of OSM extended with time intervals is similar to the standard valid-time static model described by Jensen and Snodgrass [JS99]. The behavioral and interaction models of OSM, however, extend far beyond these static models. Notationally, we represent within-object behavior in OSM by a state-transition-like diagram and between-object interactions with a message-passing-like diagram. Formally, we map both behavior and interaction to two-sorted logic predicates and temporal constraints expressed in first-order logic. These first-order expressions can represent any of the temporal operators defined by Manna and Pnueli [MP91] such as *Next* ($\bigcirc$), *Eventually* ($\Diamond$), *Since* ($\mathcal{S}$), and *Until* ($\mathcal{U}$).

Although OSM is expressively powerful, notationally convenient, and useful in analyzing real-world situations and specifying systems, its reasoning properties have not been fully established (i.e., its ability to check satisfiability, subsumption, and entailment). OSM-Logic, however, is conceptually similar to description logics [CLN98], and we show in the appendix that OSM-Logic maps into $\mathcal{DLR}_{\mathcal{US}}$ [AF09], a temporal description logic. The mapping takes OSM-Logic constructs to similar $\mathcal{DLR}_{\mathcal{US}}$ constructs and takes OSM-Logic temporal constraints to the *Since* ($\mathcal{S}$) and *Until* ($\mathcal{U}$) operators of $\mathcal{DLR}_{\mathcal{US}}$. Thus, since Artale and Franconi [AF09] are able to conclude that "the fragment, $\mathcal{DLR}^-_{\mathcal{US}}$, of $\mathcal{DLR}_{\mathcal{US}}$ deprived of the ability to talk about temporal persistence of n-ary relations, for $n \geq 2$, is decidable" and that "reasoning in $\mathcal{DLR}^-_{\mathcal{US}}$ is an EXPTIME-complete problem," then OSM-Logic without temporality for n-ary relationship sets ($n > 2$) is decidable and EXPTIME-complete. We leave a full determination of the reasoning properties for OSM-Logic to future work.

In the remainder of the chapter, we give the details of OSM-Logic. Section 2 describes OSM-Logic itself, Section 3 describes how we attach semantics to a set of formulas, Section 4 summarizes the OSM-to-OSM-Logic conversion algorithm, and we conclude in Section 5.

2 OSM-Logic Language Definition

OSM-Logic is a multi-sorted language with two basic sorts, $S = \{s_t, s_o\}$; s_t is for time points and s_o is for objects. OSM-Logic consists of an infinite set of symbols arranged as prescribed in [End72]. *Logic symbols* include *logical connectors*, $\vee$, $\Rightarrow$, $\neg$; *time-variable symbols*, V_t (e.g., t_1, t_2, ...); *object-variable symbols*, V_o; *equality symbols*, $=_t$ for s_t and $=_o$ for s_o; and *auxiliary symbols*, parentheses and comma. *Parameters* include *quantifiers*, $\{\exists_t, \forall_t\}$ for s_t and $\{\exists_o, \forall_o\}$ for s_o;[1] *time-constant symbols*, C_t, *object-constant symbols*, C_o; *predicate symbols* of sort $\langle s_1, ..., s_n \rangle$, where $s_j \in S$ for $1 \leq j \leq n$[2]; and n-place *function symbols* of sort $\langle s_1,\ ...,\ s_n, s_o \rangle$, where $s_j \in S$ for $1 \leq j \leq n$. Note that we have restricted the results of functions to be objects.

Terms with sort s_t include time-variable and time-constant symbols ($V_t \cup C_t$). Terms with sort s_o include object-variable and object-constant symbols ($V_o \cup C_o$) along with function terms, which we construct from function symbols by filling each place of the symbol with a term of the designated basic sort. For example, if $+$ is a 2-place function symbol of sort $\langle s_o,\ s_o,\ s_o \rangle$ and x and y are object-variable symbols in V_o, we construct a function term $+$ written either $+(x, y)$ (prefix notation) or $x + y$ (infix notation).

An *atomic formula* is a sequence $P(z_1,\ ...,\ z_n)$, where P is an n-place predicate symbol or an equality symbol (in which case $n = 2$) of sort $\langle s_1,\ ...,\ s_n \rangle$ and z_1, ..., z_n are terms of sort s_1, ..., s_n, respectively. For example, let *Pizza*(_, _, _) be a 3-place predicate symbol of sort $\langle s_o,\ s_t,\ s_t \rangle$ that represents the membership of the *Pizza* object set in Figure 1. Also, let x be an object-variable symbol and t_1 and t_2 be time-variable symbols. We can construct the atomic formula *Pizza*(x, t_1, t_2). We often write atomic formulas using an infix notation, as with the general constraint in Figure 1 written $x + y > 1$ rather than $> (x + y, 1)$.

A *well-formed formula* (wff) is constructed from atomic formulas, logical connectors, and quantifiers in the traditional way. Figure 2 shows two wff's constructed from atomic formulas, the $\Rightarrow$ logical connector, and universal quantifiers. In the first formula, *Crust*(_) *is subpart of Pizza*(_)(_, _) is a predicate symbol of sort $\langle s_o,\ s_o,\ s_t,\ s_t \rangle$ that represents the membership of the *Crust is subpart of Pizza* relationship set in Figure 1. To aid readability, we write the object terms for this predicate symbol in-line, using an infix notation. The second predicate symbol in the first formula, *Crust*(_, _, _), represents the memberships of the *Crust* object set in Figure 1. Informally, the first formula guarantees that an object is a member of the *Crust* object set whenever it relates to a pizza in the *Crust is subpart of Pizza* relationship set. Similarly, the second formula guarantees that an object is a member of the *Pizza* object set whenever it relates to a crust in the *Crust is subpart of Pizza* relationship set.

3 Interpretations

We establish the meaning of a formula or a set of formulas through an *interpretation*.

[1] When the sort is clear from the context, we may drop the t or o subscript from an equality symbol or quantifier.

[2] However, no predicate symbol has more than two places of sort s_t.

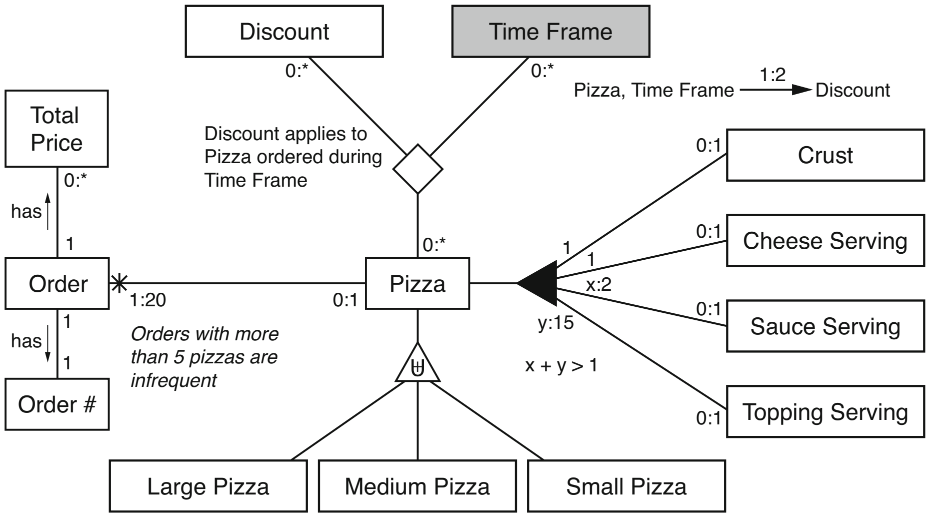


Fig. 1. Sample ORM for Pizza Ordering System

$$\forall x \forall y \forall t_1 \forall t_2 (\mathit{Crust}(x) \ \mathit{is\ subpart\ of}\ \mathit{Pizza}(y)(t_1, t_2) \Rightarrow \mathit{Crust}(x, t_1, t_2))$$
$$\forall x \forall y \forall t_1 \forall t_2 (\mathit{Crust}(x) \ \mathit{is\ subpart\ of}\ \mathit{Pizza}(y)(t_1, t_2) \Rightarrow \mathit{Pizza}(y, t_1, t_2))$$

Fig. 2. Sample OSM-Logic Formulas

An interpretation maps the language's parameter symbols to a mathematical structure, consisting of a time structure, a universe of objects, a set of functions, and a set of relations. As a result, an interpretation gives meaning to the symbols of the language. Without an interpretation, a formula is just a sequence of symbols and nothing more. For example, the formulas shown in Figure 2 are by themselves just sequences of symbols. An interpretation gives them meaning by mapping $\mathit{Crust}(\_)$ *is subpart of* $\mathit{Pizza}(\_)(\_, \_)$, and $\mathit{Pizza}(\_, \_)$ to relations, x and y to objects, and t_1 and t_2 to time points.

Formally, we define an *interpretation* to be an 8-tuple $\langle T, U, F, R, g_T, g_U, g_F, g_R\rangle$ where T, U, F, and R form the mathematical structure and g_T, g_U, g_F, and g_R map parameter symbols to elements of T, U, F, and R, respectively:

- T is a time structure such that it includes (1) a (possibly infinite) set of time points TP, (2) a total ordering $<$ on TP, and (3) a time-interval magnitude function $f_{||} : TP \times TP \to U$, such that $f_{||}(\tau_1, \tau_1) = 0$, $f_{||}(\tau_1, \tau_2) = f_{||}(\tau_2, \tau_1)$, and $f_{||}(\tau_1, \tau_2) < f_{||}(\tau_1, \tau_3)$ for $\tau_1 < \tau_2 < \tau_3$.
- U is a non-empty universe of objects.
- F is a set of functions such that it includes the time-interval magnitude function. Each function in F of arity n has a sort $\langle s_1, ..., s_n, s_o\rangle$ where $s_j \in S$ for $1 \le j \le n$. The time-interval magnitude function has arity 2 and sort $\langle s_t, s_t, s_o\rangle$.

R is a set of relations such that it includes the $<$ ordering relation. Each relation of arity n has a sort $\langle s_1, ..., s_n \rangle$, where $s_j \in S$ for $1 \leq j \leq n$. The $<$ relation has arity 2 and sort $\langle s_t, s_t \rangle$.

g_T is a mapping of time constant symbols to TP.

g_U is a mapping of object constant symbols to U.

g_F is a mapping of function symbols to F such that it maps an n-place function symbol to an n-ary function of the same sort.

g_R is a mapping of predicate symbols to R such that it maps an n-place predicate symbol to an n-ary relation of the same sort.

To give OSM-Logic temporal semantics, we add three restrictions to the definition of an interpretation. First, relations in R include exactly zero, one, or two arguments of the time sort. We call these respectively *time-invariant*, *event*, and *temporal relations*. The two time points in a tuple from a temporal relation identify a *time interval* over which the other objects in the tuple are related. Let τ_1 and τ_2 be time points. If $\tau_1 < \tau_2$, then the time interval $[\tau_1, \tau_2)$ is the set of time points $t \in TP$ such that $\tau_1 \leq t < \tau_2$. If $\tau_2 < \tau_1$ then the time interval $[\tau_1, \tau_2)$ is the set of time points t, such that $\tau_2 \leq t < \tau_1$. If $\tau_1 = \tau_2$, then the time interval $[\tau_1, \tau_2)$ is the empty set. By definition, the time interval $[\tau_1, \tau_2)$ is the same as $[\tau_2, \tau_1)$. The notation "[...)" reminds us that the interval includes the starting point but not the ending point.

Second, we restrict temporal relations so if a temporal relation includes a tuple that relates a set of objects for some time interval, then it also includes tuples that relate the same set of objects for all non-empty sub-intervals of that time interval. Let R be an n-ary temporal relation R. If $(x_1, ..., x_n, t_1, t_2) \in R$ and there exists $t_3 \in TP$ such that $t_1 < t_3 < t_2$, then $(x_1, ..., x_n, t_1, t_3) \in R$ and $(x_1, ..., x_n, t_3, t_2) \in R$. The temporal relations r_a, r_b, and r_c in Figure 3 satisfy this restriction.

Third, we restrict temporal relations so if a temporal relation includes two tuples with the same set objects and the tuples have adjacent or overlapping time intervals, then the relation must also contain a third tuple with the same objects and a time interval that spans both of the others. Two time intervals, $[t_1, t_2)$ and $[t_3, t_4)$, are *adjacent* if $t_2 = t_3$ or $t_1 = t_4$ and *overlapping* if $t_1 < t_3 < t_2 < t_4$ or $t_3 < t_1 < t_4 < t_2$. We formally define this restriction as follows. Let R be an n-ary temporal relation R. If $(x_1, ..., x_n, t_1, t_2)$, $(x_1, ..., x_n, t_3, t_4) \in R$ and the time intervals $[t_1, t_2)$ and $[t_3, t_4)$ are either adjacent or overlapping, then $(x_1, ..., x_n, t_5, t_6) \in R$ where $t_5 = t_1$ and $t_6 = t_4$ if $t_1 < t_3$, otherwise $t_5 = t_3$ and $t_6 = t_2$. The temporal relations r_a, r_b, and r_c Figure 2 also satisfy this restriction.

Given an interpretation, a formula is either true or false. For example, consider the interpretation I in Figure 3. Note that I maps the *Crust*(_) *is subpart of Pizza*(_)(_, _) predicate symbol to r_a, the *Crust*(_, _, _) predicate symbol to r_b, and the *Pizza*(_, _, _) predicate symbol to r_c. Using I, the first formula in Figure 2 is true, because every object $x \in U$, time $t_1 \in TP$, and time $t_2 \in TP$ that is associated in the first, third, and fourth places of r_a, respectively, is also associated in r_b. The second formula in Figure 2 is also true, because every

$I = \langle T, U, F, R, g_T, g_U, g_F, g_R \rangle$, where

$T = \{\ TP = \{1, 2, 3, 4, 5\}$, the $<$ ordering relation, and the time-interval magnitude function $f_{||}$, where $f_{||}(\tau_1, \tau_2)$ is the absolute value of $\tau_2 - \tau_1$
$\}$

$U = \{p_1, p_2, p_3, c_1, c_2, 0, 1, 2, 3, 4\}$

$F = \{f_{||}\}$

$$R = \{ \begin{array}{cccc} \multicolumn{4}{c}{r_a} \\ \hline c_1 & p_1 & 3 & 4 \\ c_2 & p_2 & 3 & 5 \\ c_2 & p_2 & 3 & 4 \\ c_2 & p_2 & 4 & 5 \\ & & & \\ & & & \\ & & & \\ & & & \\ & & & \\ & & & \end{array}, \begin{array}{ccc} \multicolumn{3}{c}{r_b} \\ \hline c_1 & 2 & 4 \\ c_1 & 2 & 3 \\ c_1 & 3 & 4 \\ c_2 & 3 & 5 \\ c_2 & 3 & 4 \\ c_2 & 4 & 5 \\ c_3 & 4 & 5 \\ & & \\ & & \\ & & \end{array}, \begin{array}{ccc} \multicolumn{3}{c}{r_c} \\ \hline p_1 & 3 & 4 \\ p_2 & 3 & 5 \\ p_2 & 3 & 4 \\ p_2 & 4 & 5 \\ & & \\ & & \\ & & \\ & & \\ & & \\ & & \end{array}, \begin{array}{cc} \multicolumn{2}{c}{<} \\ \hline 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \\ 2 & 3 \\ 2 & 4 \\ 2 & 5 \\ 3 & 4 \\ 3 & 5 \\ 4 & 5 \end{array} \}$$

$g_T = \{\}$

$g_U = \{\}$

$g_F = \{\langle ||, f_{||} \rangle\}$

$g_R = \{\ \langle \mathit{Crust}(\_)\ \mathit{is\ subpart\ of\ Pizza}(\_)(\_, \_), r_a \rangle,$
$\langle \mathit{Crust}(\_, \_, \_), r_b \rangle,$
$\langle \mathit{Pizza}, r_b \rangle$
$\}$

Fig. 3. An Interpretation for the Formulas in Figure 2

object $x \in U$, time $t_1 \in TP$, and time $t_2 \in TP$ that are associated in the second, third, and fourth places of r_a, respectively, are also associated in r_c.

If an interpretation I makes a formula α true, then I is called a *valid interpretation*[3] for α. If I makes each formula in a set of formulas true, then I is a valid interpretation for the set of formulas. The interpretation in Figure 3 is a valid interpretation for the set of formulas in Figure 2.

The semantics of a set of formulas Γ is defined by all the possible valid interpretations for Γ. Intuitively, we think of a set of formulas in a logic language as a declaration about the characteristic properties of a system. Valid interpretations formalize this notion by defining all the possible situations that reflect these characteristic properties. For example, consider a system described by the formulas in Figure 2. The characteristic properties of this system are that whenever an object is in the first place of r_a, it must also be in the first place of r_b, and whenever an object is the second place of r_a, it must also be in the first place of r_b. In other words, pizzas can relate only to crust objects and crust objects can relate only to pizza objects in *Crust is subpart of Pizza*. The set of all valid interpretations for these formulas represents precisely all situations that reflect these characteristic properties.

[3] "Model" is the usual logic term, but since "model" is heavily overloaded in the context of computer science, we use the term "valid interpretation" instead.

4 OSM-to-OSM-Logic Conversion Algorithm

Using OSM-Logic we can now formally define the semantics of OSM by algorithmically converting any OSM model instance to a set of OSM-Logic formulas. Using an interpretation, we can then give meaning to the set of formulas by mapping their symbols to a mathematical structure. The mathematical structures of the valid interpretations for these formulas represent all possible situations that exhibit the characteristic properties described by the model instance.

The OSM-to-OSM-Logic conversion algorithm consists of a set of preliminary transformations and a set of independent conversion procedures. The preliminary transformations simply convert an OSM model instance to a standard form (e.g., standardizing names and giving identifiers to unnamed elements). Each conversion procedure generates OSM-Logic formulas for a particular type of modeling component or modeling construct that links components together in a specific way. In Sections 4.1 and 4.2 we summarize these procedures for OBMs and OIMs respectively using the pizza example (Figure 1). Because the details are extensive, we only summarize here; the full conversion algorithm is provided in [CEW92].

We outline the conversion procedures in Figures 4, 5, and 6. We organize the conversion procedures into sub-procedures for converting static and dynamic properties in each of the sub-model types.

Static Properties

1. Ensure referential integrity of relationship sets
2. Represent generalization/specialization relationship sets
3. Ensure generalization/specialization constraints
4. Represent aggregation relationship sets
5. Represent participation constraints
6. Represent co-occurrence constraints
7. Map variables used in constraints
8. Represent notes
9. Represent general constraints

Dynamic Properties

1. Represent “becoming” and “ceasing-to-be” for object sets
2. Represent “becoming” and “ceasing-to-be” for relationship sets

Fig. 4. Summary of ORM-to-OSM-Logic Conversion Procedures for Pizza Example

For example, the ORM conversion procedure (Figure 4) includes nine sub-procedures for mapping static properties of ORMs into OSM-Logic, and two for for mapping dynamic properties. The first static-property procedure generates formulas that ensure the referential integrity of relationship sets. If an object o is involved in the i^{th} position of a tuple in some relationship set R, then o must be a member of the object set associated with the i^{th} connection of R. The formulas in Figure 2 illustrate this: the first guarantees that if $\langle x, y, t_1, t_2 \rangle$ is a tuple in *Crust is subpart of Pizza*, then x must be a member of *Crust*, and the second similarly

Static Properties

1. Ensure referential integrity of states
2. Ensure referential integrity of transitions
3. Ensure referential integrity of real-time markers
4. Represent state conjunctions
5. Represent prior-state conjunction reset action
6. Represent transition firing phase
7. Ensure mutual exclusion of transition states
8. Ensure that objects are in at least one phase for each transition
9. Ensure that objects do not commit conflicting transitions
10. Represent real-time markers
11. Represent real-time constraint semantics

Dynamic Properties

1. Represent transition ready phase conditions
2. Represent transition committed phase conditions
3. Represent transition execution phase conditions
4. Represent transition finishing phase conditions
5. Represent transition inactive phase conditions
6. Specify when objects can enter each state
7. Specify when objects can exit each state

Fig. 5. Summary of OBM-to-OSM-Logic Conversion Procedures for Pizza Example

Static Properties

1. Ensure referential integrity of temporal relations
2. Ensure that a single interaction only occurs once
3. Represent source/destination parameter exchange

Dynamic Properties

1. Guarantee alignment of interaction with object time intervals

Fig. 6. Summary of OIM-to-OSM-Logic Conversion Procedures for Pizza Example

ensures the referential integrity of *Pizza* for the same relationship set. These formulas are true for all time intervals. Apart from the addition of temporal semantics, the formulas in Figure 2 are straightforward and correspond to what one would expect for structural properties. We omit discussion of procedures 2–9 for ORM static properties because they are similarly straightforward.

However, converting dynamic properties with their full temporal formalisms is a more interesting and less common problem that requires more explanation. Formulas representing dynamic properties have time-dependent membership conditions and are not true for all time intervals. In real-world systems, classification and relationships change over time, so objects and relationships become and cease to be members of object and relationship sets over time as well. Thus, we must write formulas that capture the notions of "becoming" and "ceasing-to-be" formally over time intervals. For each object and relationship set we generate additional predicates (e.g., *Becoming Pizza*(_, _, _) of sort $\langle s_o, s_t, s_t\rangle$ and *Ceasing to be Crust*(_) *is subpart of Pizza*(_)(_, _) of sort $\langle s_o, s_o, s_t, s_t\rangle$) to represent these properties.

The conversion routine generates formulas like the following to capture the semantics of these dynamic properties:

$$\begin{aligned}&\forall x \forall t_1 \forall t_2 (\mathrm{STI}(\mathit{Pizza}(x, t_1, t_2), t_1, t_2) \Rightarrow \\ &\qquad \exists t_3 (t_1 < t_3 \wedge \mathit{Becoming\ Pizza}(x, t_1, t_3)))\end{aligned} \tag{1}$$

where $\mathrm{STI}(F, t_1, t_2)$ is shorthand notation for a starting-time interval formula defined as $(t_1 < t_2 \wedge F \wedge \forall t_3 (t_3 < t_1 \Rightarrow \neg F_{t_3}^{t_1}) \wedge \exists t_3 (t_3 < t_1))$. Formula 1 ensures that the time intervals are aligned such that when an object starts being a member of the *Pizza* object set, it was in the process of becoming a pizza. Other generated formulas ensure that all the required conditions relating the alignment of various dynamic time intervals hold true. There are eight such conditions [CEW92]. The conversion procedure generates similar formulas for relationship sets as well.[4]

4.1 Converting OBM Components

The procedures that convert the OBM components to OSM-Logic define the static and dynamic behavioral properties for members of object sets. Static behavioral properties are object-set membership conditions involving object behavior represented by states, transitions, and real-time markers. Like the static properties represented in an ORM instance, static behavioral properties for an object set are conditions that must be satisfied for all time intervals. Dynamic behavioral properties are object-set membership conditions that relate various time intervals represented by OBM components.

Since an OBM instance is a collection of state nets and each state net describes object behavior for a single object set, we can execute the OBM conversion procedures independently for each state net. Therefore, without the loss of generality, we simplify our definition of the OBM conversion procedures by describing them for a single state net. Using the state net shown in Figure 7 as an example, we summarize the OBM conversion procedures in the next two subsections.

Static Properties. To represent the static behavioral properties described in a state net, we construct predicates for the events and temporal relations represented by various components in the state net, including: states, state conjunctions, transitions, triggers, actions, and real-time markers. We use component names and identifiers, together with an object-set name for the associated object set, to construct these predicates using a set of templates as Figure 8 shows. The template parameters are defined as follows: $\langle$ObjectSet$\rangle$ is an object-set name (e.g., *Order*). $\langle$SC$\rangle$ is a state name or a conjunction of state names (e.g., *Paid*

[4] We also define shorthand notations for ending- and maximum-time intervals. $\mathrm{ETI}(F, t_1, t_2)$ represents a formula that is true iff t_1 and t_2 represent an ending-time interval for F, regardless of the interpretation. Similarly, $\mathrm{MTI}(F, t_1, t_2)$ is true iff t_1 and t_2 represent a maximum-time interval.

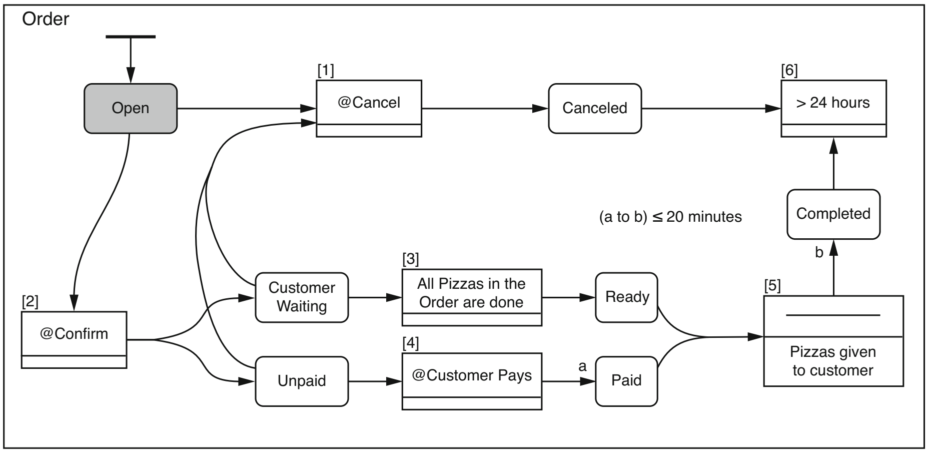


Fig. 7. State Net for the *Order* Object Set

Predicate	Sort
⟨ObjectSet⟩(_) *in state* ⟨SC⟩(_, _)	$\langle s_o, s_t, s_t \rangle$
⟨ObjectSet⟩(_) *transition* ⟨TID⟩ ⟨phase⟩(_, _)	$\langle s_o, s_t, s_t \rangle$
⟨ObjectSet⟩(_) *transition* ⟨TID⟩ *trigger true*(_, _)	$\langle s_o, s_t, s_t \rangle$
⟨ObjectSet⟩(_) *transition* ⟨TID⟩ *committed using* ⟨PSC⟩(_)	$\langle s_o, s_t \rangle$
⟨ObjectSet⟩(_) ⟨PSC⟩ *reset*(_, _)	$\langle s_o, s_t, s_t \rangle$
⟨ObjectSet⟩(_) *transition* ⟨TID⟩ *action done*(_)	$\langle s_o, s_t \rangle$
⟨ObjectSet⟩(_) *transition* ⟨TID⟩ *finished using* ⟨SSC⟩(_)	$\langle s_o, s_t \rangle$
⟨ObjectSet⟩(_) *passed* ⟨RTM⟩ *at time*(_)	$\langle s_o, s_t \rangle$

Fig. 8. Predicate Templates for OBM Components

or *Ready & Paid*). ⟨TID⟩ is a transition identifier (e.g., *[1]*) and ⟨RTM⟩ is a real-time marker (e.g., *a* or *b*). ⟨PSC⟩ and ⟨SSC⟩ are prior-state conjunctions (e.g., *Ready & Paid*) and subsequent-state conjunctions (e.g., *Customer Waiting & Unpaid*) respectively. ⟨phase⟩ is one of *inactive*, *ready*, *committed*, *executing*, and *finishing*. When a transition is committed, executing, or finishing, we say that it is *firing*, and so we also include a sixth *firing* phase that is equivalent to the disjunction of the other three.

The templates in Figure 8 each include one symbol of sort s_o and one or two symbols of sort s_t because all the predicates relate objects in an object set to events (e.g., finishing a transition) or time intervals (e.g., being in a state) associated with an OBM model instance.

Using the generated predicate symbols, we construct formulas for the static behavioral properties. In all, there are 23 conversion procedures that generate formulas. However, only 11 of them produce formulas for our sample model instance. The first three generate referential-integrity formulas, which we do not show here. They guarantee, for example, that when an object is in the *Open* state, it is a member of the *Order* object set, etc.

A fourth procedure generates formulas like Formula 2 to guarantee equivalence between a state conjunction and the conjunction of its states.

$$\begin{aligned} &\forall x \forall t_1 \forall t_2 ((\mathit{Order}(x)\ \mathit{in\ state\ Ready}(t_1, t_2) \wedge\ \mathit{Order}(x)\ \mathit{in\ state\ Paid}(t_1, t_2)) \\ &\qquad \Leftrightarrow \mathit{Order}(x)\ \mathit{in\ state\ Ready\ \&\ state\ Paid}(t_1, t_2)) \end{aligned} \tag{2}$$

A fifth procedure generates formulas like Formula 3 that specify what it means for a prior-state conjunction to be reset (i.e., an object uses the prior-state conjunction to commit a transition).

$$\begin{aligned} &\forall x \forall t_1 \forall t_2 ((\mathrm{NTI}(\mathit{Order}(x)\ \mathit{in\ state\ Ready}(t_1, t_2),_t 1_t 2) \wedge \\ &\qquad \mathrm{NTI}(\mathit{Order}(x)\ \mathit{in\ state\ Paid}(t_1, t_2), t_1, t_2)) \Leftrightarrow \\ &\qquad \mathit{Order}(x)\ \mathit{Ready\ \&\ Paid\ reset}(t_1, t_2)) \end{aligned} \tag{3}$$

Here, NTI($\mathit{Order}(x)$ *in state Ready*$(t_1, t_2), t_1, t_2$) is a shorthand for the following formula that is true if and only if x is not in the *Ready* state for all time points in $[t_1, t_2)$ time interval.

$$\begin{aligned} &t_1 < t_2 \wedge \forall t_3 \forall t_4 ((t_1 \leq t_3 \wedge t_3 \leq t_2 \wedge t_1 \leq t_4 \wedge t_4 \leq t_2) \Rightarrow \\ &\qquad \neg \mathit{Order}(x)\ \mathit{in\ state\ Ready}(t_3, t_4)) \end{aligned} \tag{4}$$

Similarly, NTI($\mathit{Order}(x)$ *in state Paid*$(t_1, t_2), t_1, t_2$) represents a formula that is true if and only if x is not in the *Paid* state for all time points in time interval $[t_1, t_2)$. Like the STI notation, an NTI shorthand can be substituted with the formula it represents without changing the meaning of the original formula.

A sixth procedure generates formulas that equate the firing phase of each transition with the concatenation of its committed, executing, and finishing phases. A seventh procedure generates formulas that guarantee the elementary phases of a transition are mutually exclusive. An eighth procedure generates formulas to guarantee that objects in the associated object set are in at least one phase of each transition in a state net. We do not give examples here.

A ninth procedure generates formulas like Formula 5 to guarantee that objects never commit conflicting transitions with common prior states at the same time. Two transitions are conflicting if they share prior states (e.g., transitions *[1]* and *[3]* conflict w.r.t. *Customer Waiting*).

$$\begin{aligned} &\forall x \forall t (\mathit{Order}(x)\ \mathit{transition}\ [1]\ \mathit{committed\ using} \\ &\qquad \mathit{Customer\ Waiting\ \&\ Unpaid}(t) \Rightarrow \\ &\qquad \neg \mathit{Order}(x)\ \mathit{transition}\ [3]\ \mathit{committed\ using\ Customer\ Waiting}(t)) \end{aligned} \tag{5}$$

A tenth procedure generates formulas like Formula 6 that restrict temporal relations represented by the predicate symbols for real-time markers so that they only relate objects to time points for which the objects passed to the real-time markers.

$$\forall x \forall t(\mathit{Order}(x)\ \mathit{passed\ a\ at\ time}(t) \Leftrightarrow \\ \mathit{Order}(x)\ \mathit{transition}\ [4]\ \mathit{finished\ using\ Paid}(t)) \tag{6}$$

Formula 6 specifies that if an object passes real-time marker a at time t, that object finishes transition *[4]* and enters the *Paid* state at time t and the converse. Thus the temporal relation represented by the predicate symbol $\mathit{Order}(\_)$ $\mathit{passed\ a\ at\ time}(\_)$ is equivalent to the temporal relationship represented by the predicate symbol $\mathit{Order}(\_)$ *transition [1] finished using* $\mathit{Paid}(\_)$.

Finally, an eleventh procedure generates formulas like Formula 7 to express real-time constraints. The real-time constraint, *(a to b)* $\leq$ *20*, in Figure 7 is an a OSM-Logic Formula, where *to* and $\leq$ are predicate symbols and a and b are variables representing values for real-time markers.

$$\forall x \forall t_1 \forall t_2((\mathit{Order}(x, t_1, t_2) \wedge t_1 < t_2 \wedge \mathit{Order}(x)\ \mathit{passed\ a\ at\ time}(t_1) \wedge \\ \mathit{Order}(x)\ \mathit{passed\ b\ at\ time}(t_2) \wedge \forall t_3((t_1 < t_3 \wedge t_3 < t_1) \Rightarrow \\ (\neg \mathit{Order}(x)\ \mathit{passed\ a\ at\ time}(t_3) \wedge \neg \mathit{Order}(x)\ \mathit{passed\ b\ at\ time}(t_3)))) \Rightarrow \\ (t_1\ \mathit{to}\ t_2) \leq \text{“20”}) \tag{7}$$

The last part of Formula 7 is derived from the real-time constraint. Here, as in Figure 7, “20” represents 20 minutes. An interpretation of this formula would map this constant symbol to an appropriate object in the range of the time-interval magnitude function. As a result, *minutes* is normalized to the time unit represented in the interpretation. The *to* symbol in the (t_1 *to* t_2) formula represents the time-interval magnitude function.

Dynamic Properties. The dynamic behavioral properties are conditions that relate various time intervals represented by OBM components. We use the predicate symbols described in the prior section to express dynamic behavioral properties in OSM-Logic. There are 12 conversion procedures that generate formulas for these properties, 7 of which generate formulas for our sample model instance. We present these 7 procedures here. One procedure generates formulas that relate certain time intervals associated with an object and the ready phase of a transition to other time intervals that occur just before, during, and just after the ready phase. For example, the procedure generates the following formulas for transition *[1]*.

$$\begin{aligned}&\forall x \forall t_1 \forall t_2 (\mathrm{STI}(\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{ready}(t_1, t_2), t_1, t_2) \Rightarrow \\ &\qquad \exists t_3 (t_3 < t_1 \wedge (\mathrm{NTI}(\mathit{Order}(x, t_3, t_1), t_3, t_1) \vee \\ &\qquad\qquad \mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{inactive}(t_3, t_1))))\end{aligned} \tag{8}$$

$$\begin{aligned}&\forall x \forall t_1 \forall t_2 (\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{ready}(t_1, t_2) \Rightarrow \\ &\qquad (\mathit{Order}(x)\ \mathit{in\ state\ Open}(t_1, t_2) \vee \\ &\qquad \mathit{Order}(x)\ \mathit{in\ state\ Customer\ Waiting}\ \&\ \mathit{state\ Unpaid}(t_1, t_2)))\end{aligned} \tag{9}$$

$$\begin{aligned}&\forall x \forall t_1 \forall t_2 (\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{ready}(t_1, t_2) \Rightarrow \\ &\qquad \mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{trigger\ true}(t_1, t_2))\end{aligned} \tag{10}$$

$$\begin{aligned}&\forall x \forall t_1 \forall t_2 (\mathrm{ETI}(\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{ready}(t_1, t_2), t_1, t_2) \Rightarrow \\ &\qquad \exists t_3 (t_2 < t_3 \wedge (\mathrm{NTI}(\mathit{Order}(x, t_2, t_3), t_2, t_3) \vee \\ &\qquad\qquad \mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{committed}(t_2, t_3) \vee \\ &\qquad\qquad \mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{inactive}(t_2, t_3))))\end{aligned} \tag{11}$$

Formula 8 defines what happens before an object enters the ready phase of transition *[1]*. The first part of this formula is true when $[t_1, t_2)$ is a starting-time interval for an object x being in the ready phase of transition *[1]*. In other words, t_1 is the time point when x enters the ready phase of transition *[1]*. The second part is conditional upon the first. It specifies that there is a time point t_3 before t_1 such that either x is not a member of *Order* or x is in the inactive phase of transition *[1]* during the $[t_3, t_1)$ time interval. Formulas 9 and 10 define what must be true while an object is in the ready phase of transition *[1]*. Formula 9 specifies that at least one of the transition's prior-state conjunctions must be true, and Formula 10 specifies that the transition's trigger must be true. Formula 11 defines what happens after an object is the ready phase of transition *[1]*. The first part of this formula is true when $[t_1, t_2)$ is an ending-time interval for an object x being in the ready phase of transition *[1]*. In other words, t_2 is the time point when x exits the ready phase. The second part, which is conditional upon the first, specifies that there exists a time t_3 after t_2 such that either x is not a member of *Order*, x is in the committed phase of transition *[1]*, or x is in the inactive phase of transition *[1]* during the $[t_2, t_3)$ time interval.

Another procedure generates formulas like Formulas 12–16 that define what happens before and after an object is in the committed phase of a transition. Formulas 12 through 16 specify what must be true just prior to or at the time an object enters the committed phase of transition *[1]*. Formula 12 guarantees that an object is in the ready phase of transition *[1]* just prior to starting the committed phase. Formula 13 says that if an object commits transition *[1]* using the *Open* state at time point t_1, then the object is in the ready phase of transition *[1]* and in the *Open* state just before t_1. Further, it says that the object is in the committed phase at t_1 and that the *Open* state is reset at t_1. Formula 14 guarantees that an object can commit transition *[1]* with only one of its prior-state conjunctions at a time. Formula 15 says that if $[t_1, t_2)$ is a starting-time interval for an object being in the committed phase of transition *[1]*, then that object committed the transition at time t_1 using one of its prior-state conjunctions.

Formula 16 guarantees that an object is in the executing phase of transition *[1]* just after it exits the committed phase.

$$\forall x \forall t_1 \forall t_2(\mathrm{STI}(\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{committed}(t_1,t_2),t_1,t_2) \Rightarrow \exists t_3(t_3 < t_1 \wedge \mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{ready}(t_3,t_1))) \tag{12}$$

$$\forall x \forall t_1(\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{committed\ using\ Open}(t_1) \Rightarrow \exists t_3 \exists t_4(t_3 < t_1 \wedge t_1 < t_4 \wedge \mathit{Order}(x)\ \mathit{in\ state\ Open}(t_3,t_1) \wedge \mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{ready}(t_3,t_1) \wedge \mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{committed}(t_1,t_4) \wedge \mathit{Order}(x)\ \mathit{Open\ reset}(t_1,t_4))) \tag{13}$$

$$\forall x \forall t_1(\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{committed\ using\ Open}(t_1) \Rightarrow \neg \mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{committed\ using\ Customer\ Waiting\ \&\ state\ Unpaid}(t_1)) \tag{14}$$

$$\forall x \forall t_1(\mathrm{STI}(\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{committed}(t_1,t_2),t_1,t_2) \Rightarrow (\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{committed\ using\ Open}(t_1) \vee \mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{committed\ using\ Customer\ Waiting\ \&\ state\ Unpaid}(t_1))) \tag{15}$$

$$\forall x \forall t_1 \forall t_2(\mathrm{ETI}(\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{committed}(t_1,t_2),t_1,t_2) \Rightarrow \exists t_3(t_2 < t_3 \wedge \mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{executing}(t_2,t_3))) \tag{16}$$

A third procedure generates formulas like Formulas 17–18 to require that an object is in the committed phase of transition *[1]* prior to being in the executing phase, and is in the finishing phase after being in the executing phase.

$$\forall x \forall t_1 \forall t_2(\mathrm{STI}(\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{executing}(t_1,t_2),t_1,t_2) \Rightarrow \exists t_3(t_3 < t_1 \wedge \mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{committed}(t_3,t_1))) \tag{17}$$

$$\forall x \forall t_1 \forall t_2(\mathrm{ETI}(\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{executing}(t_1,t_2),t_1,t_2) \Rightarrow \exists t_3(t_2 < t_3 \wedge \mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{finishing}(t_2,t_3))) \tag{18}$$

A fourth procedure generates formulas like Formulas 19–22 to express the dynamic properties of the finishing phase, such as how the completion of actions and the entering of subsequent-state conjunctions relate to the finishing of transitions.

$$\begin{aligned}\forall x \forall t_1 \forall t_2 (\mathrm{STI}(&\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{finishing}(t_1, t_2), t_1, t_2) \Rightarrow \\ &\exists t_3 (t_3 < t_1 \wedge \mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{executing}(t_3, t_1) \wedge \\ &\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{action\ done}(t_1)))\end{aligned} \tag{19}$$

$$\begin{aligned}\forall x \forall t_1 (&\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{finished\ using\ Canceled}(t_1) \Rightarrow \\ &\exists t_3 \exists t_4 (t_3 < t_1 \wedge t_1 < t_4 \wedge \\ &\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{finishing}(t_3, t_1) \wedge \\ &\mathit{Order}(x)\ \mathit{in\ state\ Canceled}(t_1, t_4)))\end{aligned} \tag{20}$$

$$\begin{aligned}\forall x \forall t_1 \forall t_2 (\mathrm{ETI}(&\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{finishing}(t_1, t_2), t_1, t_2) \Rightarrow \\ &\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{finished\ using\ Canceled}(t_2))\end{aligned} \tag{21}$$

$$\begin{aligned}\forall x \forall t_1 \forall t_2 (\mathrm{ETI}(&\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{finishing}(t_1, t_2), t_1, t_2) \Rightarrow \\ &\exists t_3 (\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{inactive}(t_2, t_3)))\end{aligned} \tag{22}$$

Formula 19 guarantees that an object must be in executing phase of transition *[1]* prior to be being in the finishing phase. In addition, it specifies that the time point when an object enters the finishing phase corresponds to the time point it completed the action of transition *[1]*. Formula 20 says that if an object finishes transition *[1]* and enters the *Canceled* state at time t_1, then that object is in the finishing phase of transition *[1]* prior to t_1, and it is in the *Canceled* state at t_1. Formula 21 guarantees that if $[t_1, t_2)$ is an ending-time interval for an object in the finishing phase of transition *[1]*, then that object finished transition *[1]* using the *Canceled* state at time t_2. Formula 22 guarantees that an object is in the inactive phase of transition *[1]* after it is in the finishing phase.

A fifth procedure generates formulas like Formulas 23–24 to define dynamic properties of the inactive phase of a transition.

$$\begin{aligned}\forall x \forall t_1 \forall t_2 (\mathrm{STI}(&\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{inactive}(t_1, t_2), t_1, t_2) \Rightarrow \\ &\exists t_3 (t_3 < t_1 \wedge (\mathrm{NTI}(\mathit{Order}(x, t_3, t_1), t_2, t_3) \vee \\ &\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{finishing}(t_3, t_1))))\end{aligned} \tag{23}$$

$$\begin{aligned}\forall x \forall t_1 \forall t_2 (\mathrm{ETI}(&\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{inactive}(t_1, t_2), t_1, t_2) \Rightarrow \\ &\exists t_3 (t_2 < t_3 \wedge (\mathrm{NTI}(\mathit{Order}(x, t_2, t_3), t_2, t_3) \vee \\ &\mathit{Order}(x)\ \mathit{transition}\ [\mathit{1}]\ \mathit{ready}(t_2, t_3))))\end{aligned} \tag{24}$$

Formula 23 guarantees that an object is either not a member of the *Order* object set or in the finishing phase of transition *[1]* prior to being in the inactive phase of transition *[1]*. Formula 23 specifies that an object is either not a member of the *Order* object set or in the ready phase of transition *[1]* after being in the inactive phase of transition *[1]*.

The last two procedures generate formulas like Formulas 25–26 that specify when objects can enter and exit a state. An object can enter (exit) a state only when the object uses it to finish (commit) a transition.

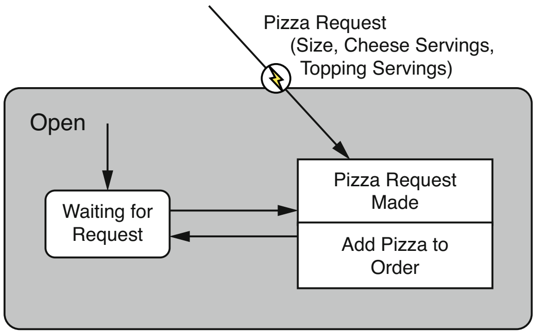


Fig. 9. Details of the Open State for the *Order* Object Set

$$\forall x \forall t_1 \forall t_2 (\text{STI}(\mathit{Order}(x) \textit{ in state } \mathit{Canceled}(t_1, t_2), t_1, t_2) \Rightarrow \\ \mathit{Order}(x) \textit{ transition } [\mathit{1}] \textit{ finished using } \mathit{Canceled}(t_1)) \tag{25}$$

$$\forall x \forall t_1 \forall t_2 (\text{ETI}(\mathit{Order}(x) \textit{ in state } \mathit{Canceled}(t_1, t_2), t_1, t_2) \Rightarrow \\ \mathit{Order}(x) \textit{ transition } [\mathit{6}] \textit{ committed using } \mathit{Canceled}(t_2)) \tag{26}$$

Formula 25 guarantees that the time point when an object starts being in the *Canceled* state corresponds to when it finishes transition *[1]* using the *Canceled* state. Similarly, Formula 26 specifies that the time point when an object stops being in the *Canceled* state corresponds to when it commits transition *[6]*.

4.2 Converting OIM Components

Static Properties. To express static interaction properties, we first construct predicate symbols for an OBM. For example, we construct the predicate symbol *Interaction Pizza Request*(_): *to Order*(_) *transition [7] firing with Size*(_) *Cheese Servings*(_) *Topping Servings*(_)(_,_) for the interaction described in Figure 9. (A preliminary transformation supplied *[7]* as the identifier for the transition in Figure 9.) The predicate symbol construction varies for each interaction component C, depending on whether C has an activity description; whether C has an origin, a destination, or both; if C has an origin, whether it is an object set, state, or transition; if C has a destination, whether it is an object set, state, or transition; and whether C has object descriptions.

All interaction predicate symbols start with *Interaction* and a place that represents interaction objects. If an interaction component has an activity description, we use that description to label the first place. For the interaction in Figure 9, we embed its activity description, *Pizza Request*, in the predicate symbol just before the first place. Next, we include places for the interaction's origin and destination. Since our sample interaction does not have an origin, we only include a place for its destination. We prefix the destination place with *to Order* to help identify it. Also, since the destination of the interaction is transition *[7]*, we embed transition *[7]* firing after the destination place to indicate that only objects firing transition *[7]* can receive this type of interaction. Next, we add a place for each object description associated with the interaction component. An object description identifies a parameter object that origin and destination objects exchange during the interaction. For our example, *Size*, *Cheese Servings*, and *Topping Servings* are the object descriptions. Finally, we add on two places of sort s_t to represent the time interval during which the interactions occur.

Using the predicate symbols generated for interaction components, objects sets, states, and transitions, we generate the formulas for static interaction properties. There are 10 conversion procedures that generate interaction formulas; however, only 3 of them pertain to our sample model instance.

One procedure generates formulas that guarantee the referential integrity of temporal relations represented by interaction predicate symbols. For example, any origin object for an interaction must be a member of the object set associated with the interaction component's origin. A second procedure generates formulas like Formulas 27–28 to ensure that a single interaction occurs only once.

$$\begin{aligned}
&\forall x_0 \forall x_1 \forall x_2 \forall x_3 \forall x_4 \forall t_1 \forall t_2 \forall t_3 \forall t_4 ((\mathrm{STI}(\mathit{Interaction\ Pizza\ Request}(x_0):\\
&\quad \mathit{to\ Order}(x_1)\ \mathit{transition}\ [7]\ \mathit{firing\ with\ Size}(x_2)\ \mathit{Cheese\ Serving}(x_3)\\
&\qquad \mathit{Topping\ Serving}(x_4)(t_1,t_2),t_1,t_2) \wedge\\
&\quad \mathit{Interaction\ Pizza\ Request}(x_0):\ \mathit{to\ Order}(x_1)\\
&\qquad \mathit{transition}\ [7]\ \mathit{firing\ with\ Size}(x_2)\ \mathit{Cheese\ Serving}(x_3)\\
&\qquad \mathit{Topping\ Serving}(x_4)(t_3,t_4)) \Rightarrow (t_1 \le t_3 \wedge t_1 \le t_4))
\end{aligned} \tag{27}$$

$$\begin{aligned}
&\forall x_0 \forall x_1 \forall x_2 \forall x_3 \forall x_4 \forall t_1 \forall t_2 \forall t_3 \forall t_4 ((\mathrm{ETI}(\mathit{Interaction\ Pizza\ Request}(x_0):\\
&\quad \mathit{to\ Order}(x_1)\ \mathit{transition}\ [7]\ \mathit{firing\ with\ Size}(x_2)\ \mathit{Cheese\ Serving}(x_3)\\
&\qquad \mathit{Topping\ Serving}(x_4)(t_1,t_2),t_1,t_2) \wedge\\
&\quad \mathit{Interaction\ Pizza\ Request}(x_0):\ \mathit{to\ Order}(x_1)\\
&\qquad \mathit{transition}\ [7]\ \mathit{firing\ with\ Size}(x_2)\ \mathit{Cheese\ Serving}(x_3)\\
&\qquad \mathit{Topping\ Serving}(x_4)(t_3,t_4)) \Rightarrow (t_3 \le t_2 \wedge t_4 \le t_2))
\end{aligned} \tag{28}$$

Formula 27 specifies that if an interaction, x_0, starts at time t_1 and it also occurs during the interval $[t_3, t_4)$, then the time points, t_3 and t_4, are no earlier than t_1. Similarly, Formula 28 says that if an interaction, x_0, ends at time t_2 and it also occurs during the interval $[t_3, t_4)$, then the time points, t_3 and t_4, are no later than t_2.

A third procedure generates formulas like Formula 29 to guarantee that source and destination objects exchange only one set of parameter objects during a single interaction.

$$\begin{aligned}
&\forall x_0 \forall x_1 \forall t_1 \forall t_2 \exists^1_{x_2} \exists^1_{x_3} \exists^1_{x_4} (\mathit{Interaction\ Pizza\ Request}(x_0):\\
&\quad \mathit{to\ Order}(x_1)\ \mathit{transition}\ [7]\ \mathit{firing\ with\ Size}(x_2)\\
&\qquad \mathit{Cheese\ Serving}(x_3)\ \mathit{Topping\ Serving}(x_4)(t_1,t_2))
\end{aligned} \tag{29}$$

Dynamic Properties. To express dynamic interaction properties we generate one or two additional predicate symbols for each interaction that represent the potential for interaction to occur. One predicate symbol, which we generate only when the interaction has an origin, relates interactions and objects to time intervals during which the objects are potential origins for the interactions. The other predicate symbol, which we generate only when the interaction has a destination, relates interactions and objects to time intervals during which the objects are potential destinations for the interactions. For example, we generate the predicate symbol *Receive Interaction Pizza Request*(_): *to Order*(_) *transition [7] firing with Size*(_) *Cheese Servings*(_) *Topping Servings*(_)(_, _), for the interaction in Figure 9.

Our algorithm includes only one procedure that generates formulas for dynamic interaction properties. The formulas guarantee that the maximum-time interval for an interaction (1) starts after a time interval corresponding to its origin object's potential to be an origin, and (2) ends before a time interval corresponding to its destination object's potential to be an destination. For example, the procedure generates the following formula for our sample model instance.

$$\begin{aligned}
&\forall x_0 \forall x_1 \forall x_2 \forall x_3 \forall x_4 \forall t_1 \forall t_2 (\\
&\quad (\mathrm{MTI}(\textit{Interaction Pizza Request}(x_0):\\
&\qquad \textit{to Order}(x_1)\ \textit{transition}\ [7]\ \textit{firing with Size}(x_2)\ \textit{Cheese Serving}(x_3)\\
&\qquad\quad \textit{Topping Serving}(x_4)(t_1, t_2), t_1, t_2) \Rightarrow\\
&\quad \exists t_3 \exists t_4 (t_1 \leq t_3 \wedge t_3 < t_2 \wedge\\
&\qquad \textit{Receive Interaction Pizza Request}(x_0):\ \textit{to Order}(x_1)\\
&\qquad\quad \textit{transition}\ [7]\ \textit{firing with Size}(x_2)\ \textit{Cheese Servings}(x_3)\\
&\qquad\quad \textit{Topping Servings}(x_4)(t_3, t_4) \wedge\\
&\qquad \forall t_5 ((t_3 \leq t_5 \wedge\\
&\qquad\quad \mathrm{ETI}(\textit{Receive Interaction Pizza Request}(x_0):\ \textit{to Order}(x_1)\\
&\qquad\qquad \textit{transition}\ [7]\ \textit{firing with Size}(x_2)\ \textit{Cheese Servings}(x_2)\\
&\qquad\qquad \textit{Topping Servings}(x_3)(t_3, t_5), t_3, t_5)) \Rightarrow t_2 \leq t_5))))
\end{aligned} \tag{30}$$

5 Concluding Remarks

We have shown how to formalize conceptualizations for applications that need time-dependent facts. The conceptualizations span the space of object existence, the existence of relationships among objects, individual object behavior, and interactions between among groups of objects. The formalization defines the semantics of OSM by showing how to convert any populated OSM model instance to formulas in OSM-Logic and how to interpret these formulas.

References

[ACM] ACM-L-2010 workshop (2010), http://www.cs.uta.fi/conferences/acm-l-2010/

[AF09] Artale, A., Franconi, E.: Foundations of Temporal Conceptual Data Models. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Mylopoulos Festschrift. LNCS, vol. 5600, pp. 10–35. Springer, Heidelberg (2009)

[CEW92] Clyde, S.W., Embley, D.W., Woodfield, S.N.: The complete formal definition for the syntax and semantics of OSA. Technical Report BYU-CS-92-2, Department of Computer Science, Brigham Young University (February 1992)

[CLN98] Calvanese, D., Lenzerini, M., Nardi, D.: Description logics for conceptual data modeling. In: Chomicki, J., Saake, G. (eds.) Logics for Databases and Information Systems, pp. 229–263. Kluwer (1998)

[Cly93] Clyde, S.W.: An Initial Theoretical Foundation for Object-Oriented Systems Analysis and Design. PhD thesis, Brigham Young University (1993)

[Dor09] Dori, D.: Object-Process Methodology: A Holistic Systems Paradigm. Springer, Berlin (2009)

[EKW92] Embley, D.W., Kurtz, B.D., Woodfield, S.N.: Object-oriented Systems Analysis: A Model-Driven Approach. Prentice-Hall, Englewood Cliffs (1992)

[End72] Enderton, H.B.: A Mathematical Introduction to Logic. Academic Press, Inc., Boston (1972)

[ET11] Embley, D.W., Thalheim, B. (eds.): Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges. Springer, Heidelberg (2011)

[HM08] Halpin, T.A., Morgan, T.: Information Modeling and Relational Databases, 2nd edn. Morgan Kaufmann, Burlington (2008)

[IAR] Knowledge discovery and dissemination program., http://www.iarpa.gov/-solicitations_kdd.html/

[JS99] Jensen, C.S., Snodgrass, R.T.: Temporal data management. IEEE Transactions on Knowledge and Data Engineering 11(1), 36–44 (1999)

[MP91] Manna, Z., Pnueli, A.: The Temporal Logic of Reactive and Concurrent Systems. Springer, New York (1991)

[Oli07] Olivè, A.: Conceptual Modeling of Information Systems. Springer, Berlin (2007)

[ORM] The ORM Foundation, http://www.ormfoundation.org

[PM07] Pastor, O., Molina, J.C.: Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling. Springer, New York (2007)

[Tha00] Thalheim, B.: Entity-Relationship Modeling: Foundations of Database Technology. Springer, Berlin (2000)

[UML] OMG: Documents associated with UML version 2.3, http://www.omg.org/-spec/UML/2.3/

[YK58] Young, J.W., Kent, H.K.: Abstract formulation of data processing problems. The Journal of Industrial Engineering 9(6), 471–479 (1958)

Appendix

Conversion of OSM to $\mathcal{DLR}_{\mathcal{US}}$

In the following we sketch the transformation of OSM into $\mathcal{DLR}_{\mathcal{US}}$ [AF09]. The transformation maps the structural components of OSM to the conceptual-modeling structures of $\mathcal{DLR}_{\mathcal{US}}$ and maps the properties of OSM listed in Figures 4, 5, and 6 to the properties of $\mathcal{DLR}_{\mathcal{US}}$. (To reference the static and dynamic properties in Figures 4, 5, and 6 we use $f.S.n$ to refer to the nth static property in Figure f and $f.D.n$ to refer to the nth dynamic property in Figure f.)

We convert an OSM model to $\mathcal{DLR}_{\mathcal{US}}$ in three steps.

First, we convert the ORM components of OSM to $\mathcal{DLR}_{\mathcal{US}}$ as follows:

1. Object sets in OSM map directly to $\mathcal{DLR}_{\mathcal{US}}$ T (*temporal, temporary, time-stamped*) classes.
2. Relationship sets in OSM map directly to T relations in $\mathcal{DLR}_{\mathcal{US}}$.
3. Referential integrity constraints (4.S.1) are converted to $\mathcal{S}$ and $\mathcal{U}$ operators in $\mathcal{DLR}_{\mathcal{US}}$.
4. Generalization/Specialization relationship sets of OSA (4.S.2) map to the *Isa* relation of $\mathcal{DLR}_{\mathcal{US}}$.
5. The generalization/specialization constraints of OSM (4.S.3) map directly to the disjointness and covering constraints of $\mathcal{DLR}_{\mathcal{US}}$.
6. The aggregation relationship sets of OSM (4.S.4) map directly to T relations in $\mathcal{DLR}_{\mathcal{US}}$.
7. Participation constraints in OSM (4.S.5) map directly to participation constraints in $\mathcal{DLR}_{\mathcal{US}}$.
8. Co-occurrence constraints in OSM (4.S.6) are converted to participation constraints on derived relations in $\mathcal{DLR}_{\mathcal{US}}$.
9. Variables in OSM constraints (4.S.7) map to variables in $\mathcal{DLR}_{\mathcal{US}}$ constraints.
10. Notes in OSM (4.S.8) are semantically meaningless and thus map to nothing in $\mathcal{DLR}_{\mathcal{US}}$.
11. The permitted general constraints of OSM (4.S.9) map directly to $\mathcal{S}$ and $\mathcal{U}$ operators in $\mathcal{DLR}_{\mathcal{US}}$. (For OSM-Logic, we restrict general constraints—allowing only those needed for $\mathcal{S}$ and $\mathcal{U}$.)
12. The “becoming” and “ceasing-to-be” constraints on object sets in OSM (4.D.1) map to $\mathcal{S}$ and $\mathcal{U}$ operators in $\mathcal{DLR}_{\mathcal{US}}$.
13. The “becoming” and “ceasing-to-be” constraints on relationship sets in OSM (4.D.2) map to $\mathcal{S}$ and $\mathcal{U}$ operators in $\mathcal{DLR}_{\mathcal{US}}$.

Second, we convert an OBM model of OSM to $\mathcal{DLR}_{\mathcal{US}}$ as follows:

1. States is OSM map to T classes in $\mathcal{DLR}_{\mathcal{US}}$.
2. Transitions in OSM map to T classes in $\mathcal{DLR}_{\mathcal{US}}$.
3. Real-time markers in OSM map to T classes in $\mathcal{DLR}_{\mathcal{US}}$.
4. The referential integrity constraints on states and transitions (5.S.1-3) map to $\mathcal{S}$ and $\mathcal{U}$ operators in $\mathcal{DLR}_{\mathcal{US}}$.

5. State conjunctions in OSM (5.S.4) map to T classes in $\mathcal{DLR}_{\mathcal{US}}$.
6. Prior-state conjunction reset actions in OSM (5.S.5) map to T classes in $\mathcal{DLR}_{\mathcal{US}}$.
7. Transition firing phases of OSM (5.S.6) map to T classes in $\mathcal{DLR}_{\mathcal{US}}$.
8. Mutual exclusion of transition states in OSM (5.S.7) maps to $\mathcal{S}$ and $\mathcal{U}$ operators in $\mathcal{DLR}_{\mathcal{US}}$.
9. Ensuring that objects are in at least one phase for each transition in OSM (5.S.8) maps to $\mathcal{S}$ and $\mathcal{U}$ operators in $\mathcal{DLR}_{\mathcal{US}}$.
10. Ensuring that objects do not commit conflicting transitions in OSM (5.S.9) maps to $\mathcal{S}$ and $\mathcal{U}$ operators in $\mathcal{DLR}_{\mathcal{US}}$.
11. Real-time markers in OSM (5.S.10) map to T classes in $\mathcal{DLR}_{\mathcal{US}}$.
12. Real-time constraint semantics in OSM (5.S.11) map to T classes in $\mathcal{DLR}_{\mathcal{US}}$.
13. All dynamic behavior properties (5.S.1-7) map to $\mathcal{S}$ and $\mathcal{U}$ operators in $\mathcal{DLR}_{\mathcal{US}}$.

Third, we convert an OIM model of OSM to $\mathcal{DLR}_{\mathcal{US}}$ as follows:

1. Sources in OIM map to T classes in $\mathcal{DLR}_{\mathcal{US}}$.
2. Destinations in OIM map to T classes in $\mathcal{DLR}_{\mathcal{US}}$.
3. Temporal relations in OIM map to T relations in $\mathcal{DLR}_{\mathcal{US}}$.
4. Referential integrity of temporal relations in OSM (6.S.1) maps to $\mathcal{S}$ and $\mathcal{U}$ operators in $\mathcal{DLR}_{\mathcal{US}}$.
5. Ensuring that single interactions only occur once in an OIM (6.S.2) maps to $\mathcal{S}$ and $\mathcal{U}$ operators in $\mathcal{DLR}_{\mathcal{US}}$.
6. Source/destination parameter exchanges in OSM (6.S.3) maps to T classes in $\mathcal{DLR}_{\mathcal{US}}$.
7. The alignment constraints of OIM (6.D.1) map to $\mathcal{S}$ and $\mathcal{U}$ operators in $\mathcal{DLR}_{\mathcal{US}}$.

Cloaking Data to Ease View Creation, Query Expression, and Query Execution

Sudarshan Murthy[1], David Maier[2], and Lois Delcambre[2]

[1] The Else Institute, 205 SE Spokane Street
Portland, OR 97202 USA
[2] Department of Computer Science, Portland State University, 1900 SW 4th Avenue
Portland, OR 97207 USA
`sudarshan.murthy@elseinstitute.org, {maier,lmd}@cs.pdx.edu`

Abstract. XML schemas often allow many aspects of an object to be described in the same document, but queries over such documents might be concerned with just one aspect. For example, an XML representation of a spreadsheet can include both spreadsheet data and styling, but a query might address only the data portions. In these situations, traditional approaches first define a *data-only view* and then query that view. However, these approaches can make it hard to define views and to express queries; and in some cases (even with view unfolding), they can even make query-execution inefficient in terms of time and space. We propose *cloaking* document parts and selectively revealing the cloaked parts as an alternative. Cloaking exposes many simultaneous conceptual views of a document without constructing new data, allows queries to be expressed in existing languages, and it can be easily supported in existing query processors. In this paper, we present a formal model for cloaking, its application in a *cloaking query processor*, and the results of an experimental evaluation in the context of *superimposed information* (SI, information with references to existing information) and *bi-level information* (SI along with the referenced information). Our experiments suggest that cloaking can make it easier to define views and to express queries, and that for certain query classes, a cloaking query processor saves both time and memory when compared to a traditional query processor.

Keywords: Views, cloaking, query optimization, bi-level information.

1 Introduction

Often, an XML document describes multiple aspects of the same object. For example, an XHTML document mixes data, metadata, styles, and scripts; an XML version of a spreadsheet might mix presentation aspects and data aspects. Such mixing can affect three common data-management activities (which we address in this paper):

- expressing aspect-specific queries which examine and return just one aspect of data,
- creating aspect-specific views, and
- executing aspect-specific queries.

We illustrate the aforementioned issues using the XML representation that Microsoft® Excel® (MS Excel) uses for spreadsheets. (Section 4 discusses these issues and our solution in the context of superimposed and bi-level information.) Figure 1 shows an XML document for a spreadsheet containing just one active cell. This document includes both the data aspect (such as cell value) and the presentation aspect (such as cell height) of the spreadsheet. The parts in bold indicate spreadsheet data; the other parts indicate presentation markup. For example, the element Styles defines three different display styles using instances of the element Style. The figure shows the full contents for only one display style. (Each Style element has five descendant elements and those elements have several attributes.)

```
<Workbook xmlns="..." xmlns:o="..." xmlns:x="..." xmlns:s="..." xmlns:h="...">
 <Styles>
  <Style s:ID="Default" s:Name="Normal">...</Style><Style s:ID="s23">...</Style>
  <Style s:ID="s30">
   <Alignment s:Vertical="Top"/>
   <Borders>
    <Border s:Position="Left" s:LineStyle="Continuous" s:Weight="2"/>
    <Border s:Position="Right" s:LineStyle="Continuous" s:Weight="2"/>
   </Borders>
   <Font s:FontName="..." x:Family="..." s:Size="11" s:Color="..."/>
  </Style>
 </Styles>
 <Worksheet s:Name="Sheet1">
  <Table s:ExpandedColumnCount="1" s:ExpandedRowCount="1" s:StyleID="s23">
   <Column s:StyleID="s23" s:AutoFitWidth="0" s:Width="91.5"/>
   <Row s:Height="15">
    <Cell s:StyleID="s30"><Data s:Type="String">Arnold Ice Cave</Data></Cell>
   </Row>
  </Table>
 </Worksheet>
</Workbook>
```

Fig. 1. XML data generated for a single MS Excel cell. Parts in bold model the cell's data; the other parts contain presentation information for the cell.

This interweaving of presentation markup with spreadsheet data makes it hard to perform the following activities involving spreadsheet data alone: expressing *data-only queries* (which examine and return only the data parts), creating data-only views, and executing data-only queries. (Likewise for *presentation-only* queries and views.)

For example, consider the task of extracting only the cell data. This task requires the following XQuery query or an equivalent XSLT template. The simple XPath expression `//Cell` does not suffice, because XPath cannot selectively remove contained nodes such as the presentation attribute s:StyleID.

```
<result>{
 for $c in doc("w.xml")//Cell return <Cell>{$c/Data}</Cell>
}</result>
```

Likewise, extracting the complete worksheet data requires a rather complex XQuery query (shown in Figure 2) due to the number and location of presentation nodes to be excluded. Again, the XPath expression `//Worksheet` does not suffice.

One way to ease expression of data-only queries is to first create a data-only view (say using the XQuery query shown in Figure 2) to expose only the spreadsheet data and execute data-only queries over that view. For example, with the data-only view created, one can use the simple XPath expression `//Cell` to easily retrieve information related to cells, without intervening presentation information. However, in addition to requiring a complex XQuery query to first create the data-only view, this approach is inefficient because the view is materialized and thus constructs new nodes (unlike XPath which simply returns existing nodes) [20].

```
<Workbook>{
  for $s in doc("w.xml")/Worksheet
   return
    <Worksheet s:Name"{s:Name}">{
     for $t in $s/Table
      return
       <Table>{
        for $r in $t/Row
         return
          <Row>{
           for $c in $r/Cell return <Cell>{$c/Data}</Cell>
         }</Row>
      }</Table>
    }</Worksheet>
}</Workbook>
```

Fig. 2. An XQuery query to create a data-only view of an MS Excel spreadsheet encoded as the XML document shown in Figure 1

Lastly, with the data of Figure 1, some data-only queries might execute inefficiently in the traditional approach. For example, executing the expression `//Cell` requires the query processor to examine all 26 elements in the document, but only six of these elements represent spreadsheet data.

One solution to these problems is to separate the spreadsheet data from its presentation markup, but this alternative organization makes it harder to express queries that operate on both presentation information and spreadsheet data. Such queries would need to explicitly join the two kinds of data, which in turn can require more complex queries in XQuery, XSLT, or XPath 2.0. Also, this organization increases the overall data-management effort because information will now be distributed.

Using indexes can address some of the query-execution problems, but typically only query processors in high-end database management systems use persistent indexes, whereas, though popular, standalone XML query processors have limited or no support for indexes. For example, Saxon [10] builds temporary indexes for very specific cases, but a user cannot decide which indexes are built and used.

In this paper, we present a means of easily and efficiently querying and returning only parts of an XML document by *cloaking* data and revealing cloaked data only to qualifying queries. Our solution has the following characteristics. It:

- creates several simultaneous conceptual views without constructing new data,
- allows user queries to be expressed in existing languages, and
- executes queries, as written, using a query processor that is largely unmodified.

The notion of cloaking is not specific to the XML model, but, for simplicity, we limit the discussion in this paper to just XML. (A detailed discussion of our solution, including application to other data models, is available elsewhere [15].)

The rest of this paper is organized as follows: Section 2 gives an overview of our solution. Section 3 generalizes the problem and presents a formal model of the solution. Section 4 introduces the notions of *superimposed information* and *bi-level information* and illustrates the benefits of cloaking in that context. It also presents the results of an experimental evaluation of a cloaking query processor. Section 5 reviews related work and Section 6 concludes the paper.

2 Solution Overview

We now informally illustrate cloaking by coloring nodes of an XPath tree [19] (corresponding to an XML document) and queries over such a tree using a simple 2-color scheme. We show how this simple scheme can be used to cloak tree nodes and then selectively reveal nodes to queries.

In the 2-color scheme, colors of tree nodes and queries are chosen from an ordered set. The tree nodes are colored such that the invariant *Color(n)* ≥ *Color(Parent(n))* is maintained for each node *n* where the ≥ relationship is performed on the ordinal position of colors in the set. Queries are not performed directly on a tree, but on the *scope* based on query color, which is a tree obtained by retaining only the nodes (and the corresponding edges) that satisfy the condition *Color(query)* ≥ *Color(node).*

The ordering of the color set and the assignment of a single color to each node are *application constraints* imposed here for illustration. As the formal model in Section 3 shows, our solution does not impose such constraints.

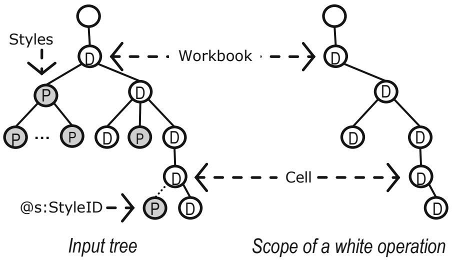


Fig. 3. A tree colored from {*White*, *Gray*}, and the scope of a white operation

In this scheme, tree nodes can be cloaked from a query by setting the node colors and the query color appropriately. For example, consider a tree whose nodes are colored from the ordered set {*White*, *Gray*}, where *Gray* > *White*. Then, the scope of a white query contains only white nodes, whereas the scope of a gray query is the entire tree.

The input tree in Figure 3 shows the XPath representation of a part of the XML document of Figure 1. The nodes of this tree are colored using the 2-color scheme. The nodes labeled D (and colored white) represent spreadsheet data; nodes labeled P (and colored gray) indicate presentation information; the unlabeled node (also colored white) is the XPath root. Some nodes are also labeled for ease of comparison with the XML document. The scope of a white query is shown on the right.

In this scheme, a white query is a data-only query because gray nodes are cloaked from it. For example, the simple XPath expression `//Cell` colored white returns the element Cell and the contained element Data, but leaves out the presentation attribute `s:StyleID`. However, if the expression is colored gray, the result includes `s:StyleID`.

A color scheme applied to an XML tree *cloaks* tree nodes, in essence defining different views over the input XML document. A query color when applied to a cloaked tree defines the *scope* of all queries of that color, creating a particular view over the original document. Figure 3 shows the scope of white queries for the example 2-color scheme applied to XML document of Figure 1. This scope (or view) is equivalent to the view created using the XQuery query of Figure 2. We note that in our approach, view creation does not create new nodes, unlike views created using XQuery or XSLT (which always create new nodes).

Cloaking also makes processing of data-only queries more efficient. For example, when executing the data-only query `//Cell` and limiting only to the nodes shown in Figure 3, a cloaking query processor needs to examine only 10 nodes (the white nodes plus the first gray child of each white node) instead of all 12 nodes as a traditional processor would. (We omit the details, but in reality, a cloaking query processor using the XPath model [19] needs to load only 44 nodes instead of loading 167 nodes.)

The 2-color scheme described thus far creates two classes of queries—data-only queries and all other queries—and it eases the task of expressing data-only queries, creating data-only views, and executing data-only queries. Likewise, a 3-color scheme employing the ordered set {*White*, *Gray*, *Slate*} and allowing multiple colors per node creates three query classes (data-only, presentation-only, and data plus presentation) and eases operations on both data-only queries and presentation-only queries. Needless to say, a 1-color scheme is the same as traditional query processing.

3 Solution Details

We now present a formal model and an architectural reference model for a cloaking query processor. We relate the architectural model to the formal model and show how the formal model applies to the relational and XML data models.

3.1 Formal Model

Formally, we model the data input to a cloaking query processor as a forest of trees. We cloak tree nodes from queries by coloring both nodes and queries from a non-empty *color set* according to a *cloaking scheme*. A *test function* determines the nodes revealed to a query based on the query's color. (Cloaking does not require colors, but we use colors to aid visualization. See the upcoming Definitions 4 and 6.)

The formal model relaxes many of the constraints of the informal model used in Section 2. For example, the formal model allows multiple colors per node, whereas the informal model assigned just one color to each node.

The formal model employs the following domains:

$\mathcal{B}$: The domain of truth values. $\mathcal{B}$ = {true, false}.
$\mathcal{D}$: The domain of colors;
$\mathcal{C}$: The domain of color sets. $\mathcal{C} = \{C \mid C \subseteq D\}$.
$\mathcal{K}$: The domain of cloaking schemes.
$\mathcal{N}$: The domain of nodes;
$\mathcal{F}$: The domain of forests.
$\mathcal{F}_k$: The domain of colored forests. $\mathcal{F}_k \subseteq \mathcal{F}$. See Definition 3.
$\mathcal{Q}$: The domain of user queries; $\mathcal{T}$: The domain of test functions. See Definition 6.

Definition 1: A *tree T* is a connected, directed acyclic graph represented by a pair *(N, E)*, where $N \subseteq \mathcal{N}$ is the set of tree nodes, and $E \subseteq (\mathcal{N} \times \mathcal{N})$ is the set of edges between the tree nodes. Each node has a structured label. The label includes the set of colors associated with the node.

Definition 2: *Colors*: $\mathcal{N} \rightarrow \mathcal{C}$ is a function that returns the colors assigned to a node. The function returns the empty set ∅ if no color is associated with the node. A cloaking scheme assigns a node's colors from a color set. See Definition 4.

Definition 3: A *forest F* is a set of trees represented by a pair *(N, E)*, where $N = \bigcup_{t=1}^{|F|} N_t$ and $E = \bigcup_{t=1}^{|F|} E_t$, where $|F|$ is the number of trees in the forest *F*. N_t and E_t denote the node set and edge set, respectively, of the t^{th} tree in the forest. A forest is *colored* if $Colors(n) \neq \emptyset, \forall n \in N$. Additionally, the forest is colored *from* the color set *C* if $Colors(n) \subseteq C, \forall n \in N$.

Definition 4: A *cloaking scheme* is a function k: $\mathcal{N} \times \mathcal{C} \times \mathcal{F} \rightarrow \mathcal{N}$ that assigns colors from a color set *C* to a node *n* in a forest *F*, producing a new node. If the nodes *n* and *k(n, C, F)* differ, it is only on their colors. Also, $Colors(k(n, C, F)) \subseteq C$.

A cloaking scheme colors each node individually, but within the context of a forest so it can examine other nodes and edges in the forest. For example, the 2-color cloaking scheme of Section 2 colors a node based on the colors of the node's parent. Another scheme might assign colors based on the tree to which the node belongs.

A cloaking scheme might impose certain constraints on the color set and the input forest. For example, the example scheme of Section 2 requires the color set to be totally ordered. Another scheme might require the forest to contain a single tree.

Definition 5: The functional *Cloak*: $\mathcal{F} \times \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{F}_k$ colors a forest $F = (N, E)$ from a color set *C* according to a cloaking scheme *k* to produce a colored forest F_k.

Cloak(F, k, C) = $F_k = (N_k, E_k)$, where:

$N_k = \{k(n, C, F) \mid n \in N\}$ and $E_k = \{(k(n_1, C, F), k(n_2, C, F)) \mid (n_1, n_2) \in E\}$

Definition 6: A *test function t*: $\mathcal{D} \times \mathcal{C} \rightarrow \mathcal{B}$ "tests" a color *c* against a color set *C*. For example, a test function might test if a query's color is one of the colors assigned to a node. Though the second input's domain is $\mathcal{C}$, in our use, its value will be a subset of the color set used to cloak the input forest. See Definition 7.

The choice of the domains $\mathcal{D}$ and $\mathcal{C}$ influences the test functions possible. For example, if the color values are unordered, a test function would be limited to equality tests on individual values. However, a function can test inequality as well if the values are ordered. (The 2-color scheme of Section 2 uses a totally ordered color set.)

Definition 7: The functional *Reveal*: $\mathcal{F}_k \times \mathcal{T} \times \mathcal{D} \rightarrow \mathcal{F}_k$ produces the scope of a query, based on query color, from a colored forest. Given a colored forest $F_k = (N_k, E_k)$ produced by *Cloak*, a test function *t*, and a query color *c*, the following holds:

$Reveal(F_k, t, c) = F_s = (N_s, E_s)$, where:

$N_s = \{n \mid n \in N_k \wedge t(c, Colors(n))\}$ and

$E_s = \{(n_1, n_2) \mid (n_1, n_2) \in E_k \wedge n_1 \in N_s \wedge n_2 \in N_s\}$

Because the set of edges E_s is equal to E_k restricted to the set of nodes in N_s, we express E_s as $E_{k \mid N_s}$ (read "E_k restricted to N_s"). Note that the revealed scope F_s might have more trees than the input colored forest F_k.

Definition 8: A *subtractor* is a function *s*: $\mathcal{F} \times \mathcal{N} \rightarrow \mathcal{B}$ that determines if a node *n* in the input forest *F* is passed to the output.

A user query can have two parts: a *subtractive* part that filters out some of the input nodes; and an *additive* part that creates new nodes, say with aggregate values. Path expressions (such as `//Cell`) in XML queries are subtractive; element and attribute constructors in XQuery and XSLT are additive. A subtractor models the subtractive part of a user query. It operates on one node at a time within the context of a forest.

Definition 9: An *adder* is a function *a*: $\mathcal{F} \rightarrow \mathcal{F}$ that produces a forest after examining an input forest. Given a query scope $F_s = (N_s, E_s)$ produced by the function *Reveal*, the following holds:

$a(F_s) = F_a = (N_a, E_a)$, where:

$N_a \subseteq \{n \mid n \in \mathcal{N} \wedge n \notin N_s\}$ and

$E_a = \{(n_1, n_2) \mid (n_1 \in N_a \wedge (n_2 \in N_a \vee n_2 \in N_s)) \vee (n_2 \in N_a \wedge (n_1 \in N_a \vee n_1 \in N_s))\}$

An adder models the additive part of a user query. It produces nodes not already in the input. (New nodes are not colored, but they can be. See Section 3.2.) If connected, a new node forms an edge with another new node or with an input node. The output forest can be ill-formed because E_a can refer to nodes not in N_a, but this situation is rectified when query execution is complete. See Definition 10.

Definition 10: The functional *Query*: $\mathcal{F}_k \times \mathcal{Q} \rightarrow \mathcal{F}_k$ computes the result of a user query over a colored forest (produced by the functional *Reveal*). Given the scope $F_s = (N_s, E_s)$ and a user query *q* with subtractive and additive parts q_s and q_a respectively, we have:

$Query(F_s, q) = F_r = (N_r, E_r)$, where
$N_r = \{n \mid n \in N_S \wedge (q_s(F_s, n) \vee n \in N_a)\}$ and $E_r = (E_s \cup E_a)_{\mid Nr}$

Here, N_a and E_a are components of the forest F_a generated by q_a, the additive part of the user query q. (The functional *Query* internally invokes q_a.)

3.2 Architectural Reference Model

Figure 4 shows a reference model for a cloaking query processor. The modules Cloak, Reveal, and Query correspond respectively to the functionals *Cloak*, *Reveal*, and *Query* in the formal model. The symbols in parentheses in Figure 4 correspond to the symbols used in Section 3.1.

Given an input forest $F = (N, E)$, a cloaking scheme k, a color set C, a test function t, a query color c, and a user query q (with subtractive and additive parts q_s and q_a), the query processor produces a forest F_r.

$F_r = (N_r, E_r) = Query(Reveal(Cloak(F, k, C), t, c), q)$, where:

$N_r = N_a \cup \{k(n, C, F) \mid n \in N \wedge t(Colors(c, k(n, C, F))) \wedge q_s(F, k(n, C, F))\}$

$E_r = E_a \cup \{(k(n_1, C, F), k(n_2, C, F)) \mid (n_1, n_2) \in E \wedge k(n_1, C, F) \in N_r \wedge k(n_2, C, F) \in N_r\}$

The equation for N_r is obtained by expanding the functionals *Query*, *Reveal*, and *Cloak* using Definitions 10, 7, and 5. New nodes created by the additive part of the user query are not colored, but they may be colored if desired. Coloring the new nodes using the same scheme used to color input nodes alters the result as follows:

$N_r = \{k(n, C, F) \mid (n \in N_a) \vee (n \in N \wedge t(Colors(c, k(n, C, F))) \wedge q_s(F, k(n, C, F)))\}$

$E_r = \{(k(n_1, C, F), k(n_2, C, F)) \mid ((n_1, n_2) \in E_a \vee (n_1, n_2) \in E) \wedge k(n_1, C, F) \in N_r \wedge k(n_2, C, F) \in N_r\}$

Regardless of the coloring of new nodes, the equations for N_r and E_r show that a cloaking query processor can execute a query without altering input nodes and without explicitly constructing the scope F_s of a user query q. Also, because the test function and the subtractor are conjunctive terms, the query processor may choose the evaluation order of these terms, possibly based on cost estimates. The processor might also be able to combine the test function with the subtractor, so that the query is executed more efficiently.

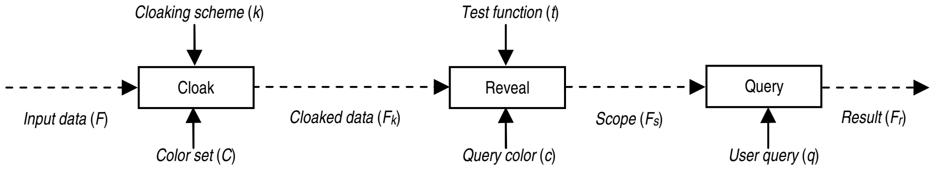


Fig. 4. An architectural reference model for a cloaking query processor. Dashed arrows indicate data flow. Solid arrows denote parameters of the query-execution process.

3.3 Key Operational Aspects

Applicable data models: Our model works for both the XML and relational data models. In the XML model, an XML document is a tree in the data models of XPath, XSLT, and XQuery. Also, XPath, XSLT, and XQuery operate on and produce trees. In the relational model, a relation instance is a tree (of height 2): The relation is the root node, a tuple in the relation is a child of the root node, and a field (that is, a column) in a tuple is a child of the tuple's node. Also, an SQL query operates on a set of trees and it outputs a single tree.

The scope of a query should be compatible with the data model and the query to be executed. These compatibility needs are met for the XML model because any scope will be a node sequence and queries in any of the three XML query languages can operate on arbitrary node sequences. With the relational model, even if the scope is a set of relations, a user query that is valid over the input data might be invalid over the scope if an entity the query explicitly references is not included in the scope. (SQL disallows referencing non-existent schema elements.) With respect to the need for the scope to be compatible with the data model, model-specific structural constraints such as "a tuple is revealed only if the containing relation is also revealed", ensures that the scope is a set of relations. (The 2-color scheme of Section 2 imposes this constraint.)

Representing and assigning colors: An input node's color can be represented at the schema level or at the instance level. Regardless, the color may be represented extensionally or intensionally; and the assignment may be implicit or explicit. Instance-level assignment affords different colors for different instances of a schema element.

Explicit extensional assignment requires support at the schema-specification level (for example, SQL's `CREATE TABLE` needs to support colors) or in a "metadata" layer. Explicit intensional assignment requires associating a query or function (that returns colors) with a schema or instance element. For example, an SQL query may be associated with a field to determine the field's color at schema or instance level.

Implicit assignment requires no additional data to represent colors, but it is limited by the information accessible to the assignment machinery. For example, an HTML node may be colored based on the kind of information it represents: data, metadata, style, script, and so on. The 2-color scheme described in Section 2 performs implicit assignment at the schema level: presentation nodes (as determined by the name and namespace of nodes) are gray; all other nodes are white.

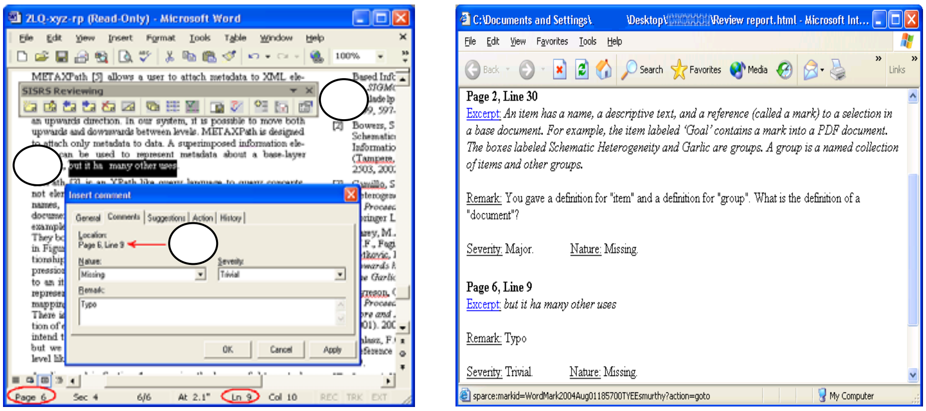
4 Evaluation

We now introduce the notions of superimposed information and bi-level information and share the results of evaluating a cloaking query processor for such information.

4.1 Superimposed and Bi-level Information

We introduce the notions of superimposed information and bi-level information using an application called the *Superimposed Scholarly Review System* (SISRS, pronounced *scissors*) [14]: SISRS is useful in peer-review processes that conferences might use. It allows reviewers to *superimpose* (that is, overlay) comments on documents and sub-documents (document parts) of various formats (PDF, HTML, MS Word, MS Excel, audio, video, and so on) without altering the base documents. Reviewers can upload superimposed comments to a central repository where an administrator can merge comments and process the merged comments in a variety of ways. For example, the administrator can filter comments based on specific attributes in the comments or by attributes in the documents reviewed. The administrator can also prepare author feedback by grouping reviews from different reviewers for each document.

Figure 5(a) shows SISRS being used, from within MS Word, to superimpose a comment on an MS Word sub-document. The Region marked "1" shows the SISRS toolbar; Region 2 shows the commented sub-document. The "Insert Comment" dialog window lets the reviewer create a structured comment. Region 3 shows the "address" (Page 6, Line 9) of the commented sub-document. SISRS automatically generates this address in both machine-friendly and human-friendly formats. Figure 5(b) shows an HTML review report generated over collated reviews. The first comment is related to content on Page 2; the second one is about Page 6 (the subject of Figure 5(a)).


2LQ-xyz-rp (Read-Only) - Microsoft Word
SISRS Reviewing
Insert comment
General | Comments | Suggestions | Action | History
OK | Cancel | Apply
Page 6 | Sec 4 | 6/6 | At 2.1" | Ln 9 | Col 10
File Edit View Favorites Tools Help
Back | Search | Favorites | Media
My Computer

Location: Page 6, Line 9
Nature: Missing
Severity: Trivial
Remark: Typo

C:\Documents and Settings\ \Desktop\ \Review report.html - Microsoft Int...

Page 2, Line 30
Excerpt: *An item has a name, a descriptive text, and a reference (called a mark) to a selection in a base document. For example, the item labeled 'Goal' contains a mark into a PDF document. The boxes labeled Schematic Heterogeneity and Garlic are groups. A group is a named collection of items and other groups.*

Remark: You gave a definition for "item" and a definition for "group". What is the definition of a "document"?

Severity: Major. Nature: Missing.

Page 6, Line 9
Excerpt: *but it ha many other uses*

Remark: Typo

Severity: Trivial Nature: Missing

sparce:markid=WordMark2004Aug01185700TYEEsmurthy?action=goto

(a) (b)

Fig. 5. (a) Using SISRS to superimpose a comment over an MS Word sub-document. (b) A review report generated via a bi-level query over review comments.

Using a *superimposed application* (SA) such as SISRS provides three key benefits: create *superimposed information* (SI) [13] in reference to *base information* (BI) without modifying BI; integrate disparate and distributed BI without replicating it; and evaluate queries over *bi-level information* [15], the combination of SI and BI.

SI is enabled by *marks* [5], which are addresses of BI. A mark can be a file-system path or a URL to a document; it can be a document address in conjunction with a fragment address such as "Page 6, Line 9" when addressing a sub-document. Obviously, what constitutes a mark depends on the location, format, and other attributes of the addressed BI. Also, multiple addressing methods might be possible for the same document or sub-document. Due to the variety of addressing needs and methods, the details of a mark are encapsulated inside a "mark descriptor" [16].

Bi-level information is SI extended by context. *Context* is a hierarchical set of information obtained from the base layer, using a mark [16]. What constitutes a context can vary across marks. For example, an MS Word sub-document has page number and line number in its context, whereas an MS PowerPoint sub-document has a slide number. Also, marks to different sub-documents within the same document can have different context elements. For example, only an MS Word sub-document located inside a table has a row number.

The notions of mark, mark descriptor, and context are supported in a middleware called SPARCE [16]. SPARCE allows pluggable mark implementations and a mark implementation may employ appropriate kinds of mark descriptors and dynamically obtain context information for a given mark descriptor.

Mark, mark descriptor, and context are also supported in *Sixml* [18], an XML representation of superimposed and bi-level information. Figures 6(a) shows the Sixml representation of the superimposed comment in Figure 5(a). The bolded parts denote SI; the other parts indicate marks and mark descriptors. The element EMark associates a mark with the SI element Comment. The element Descriptor inside EMark encodes the mark descriptor corresponding to the commented region. This descriptor identifies the mark implementation that mediates access to the base layer. It also includes the addresses of the base document and the marked sub-document.

The AMark element associates a mark with the SI attribute excerpt as denoted by the attribute target of AMark. The attribute valueSource indicates that the target attribute's run-time value is the text excerpt obtained from the associated mark. (Not shown, but both AMark and EMark contain identical Descriptor elements.)

Figures 6(b) shows an XML representation of a part of the context information obtained from the MS Word sub-document in Figure 5(a). It shows the text content of the marked sub-document; presentation information such as font name and size; and placement information such as page number and line number.

```
<Comment excerpt="">
  Typo
  <AMark target="excerpt" valueSource="true">
   <Descriptor>...<Descriptor>
  </AMark>
  <EMark>
   <Descriptor>
     <Agent>OfficeAgents.WordAgent</Agent>
     <Doc location="c:/2LQ-XYZ-rp.doc"/>
     <Subdoc startChar="395" endChar="420"/>
   <Descriptor>
  </EMark>
</Comment>
```

(a)

```
<Context>
  <Content>
    <Text>but it ha  many other uses</Text>
  </Content>
  <Presentation>
    <FontName>Times New Roman</FontName>
    <FontSize>11</FontSize>
  </Presentation>
  <Placement>
    <Page>6</Page>
    <Line>9</Page>
  </Placement>
<Context>
```

(b)

Fig. 6. (a) A Sixml representation of the superimposed comment in Figure 5(a). (b) Partial context information for the MS Word sub-document selected in Figure 5(a).

4.2 Bi-level Querying

A *bi-level query processor* evaluates *bi-level queries* (which are queries over bi-level information) [15]. Internally, a bi-level query processor represents Sixml documents using an extension of the XPath data model, but user queries are expressed in an existing XML query language. Figure 7(a) shows a simplified form of the Sixml fragment of Figure 6(a) represented in the XPath model.

The bi-level query processor performs the following tasks to ease view creation, query expression, and query execution in a bi-level-information setting:

1. Make a mark association node a child of the node with which the corresponding mark is associated. Figure 7(b) shows the AMark node as a child of the attribute excerpt, because AMark associates a mark with that attribute. This extension of the XPath data model does not affect query expression or evaluation.
2. Lazily add a node for the Context element as a child of a mark association element. Also, lazily set the value of a node to be the text excerpt retrieved using a mark if the attribute valueSource is enabled in the corresponding mark association node. Figure 7(c) illustrates this action: The AMark and EMark nodes each have a child named Context, and the value of the attribute excerpt has changed to the text that is commented on. The content of Context is omitted for brevity, but its contents will be based on the information in Figure 6(b).
3. Implicitly color tree nodes from the ordered set {*White*, *Gray*, *Slate*, *Black*} while maintaining the invariant *Color(n)* ≥ *Color(Parent(n))* for each node *n* in the tree: White to SI; Gray to mark associations (AMark and EMark); Slate to mark descriptors (Descriptor and its children); and Black to context information (Context and its children). Figure 7(c) illustrates this coloring.

The bi-level query processor's cloaking scheme supports four classes of queries: SI-only (White), SI plus mark associations (Gray); SI, mark associations, and mark descriptors (Slate); and everything (Black). Thus, as seen in Figure 7(d), a gray query sees both SI and mark associations, but a white query sees only SI (Figure 7(e)).

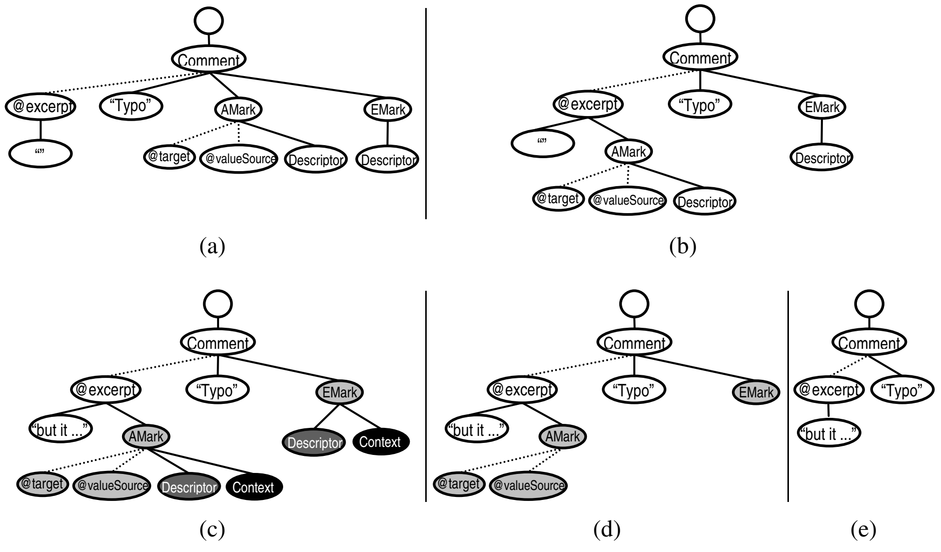


Fig. 7. (a) The Sixml document of Figure 5(a) represented in the XPath model (b) The Sixml document of Figure 5(a) represented in the extended XPath model (c) Excerpts and other context information added (d) Scope of a gray query (e) Scope of a white query

4.3 Implementation and Observations

We have implemented a bi-level query processor using the .NET Framework [17] and we have used this query processor with several SAs including SISRS. In this section, we share our observations on the ease of view creation and query expression and share some experimental results showing improved query-execution efficiency due to cloaking, all in the context of bi-level information generated by SISRS.

First, the bi-level query processor's cloaking scheme automatically defines and creates four simultaneous views (which are in turn query scopes) over bi-level information created with any SA (not just for SISRS). Users can readily access these views by simply choosing an appropriate query color. For example, Figure 7(e) shows the SI-only view over which white queries execute.

In terms of query expression, retrieving all SI requires the simple XPath expression "." colored white and executed in the context of the root node. In contrast, retrieving SI with a traditional processor requires a 96-line XSLT style sheet containing eight templates that employ 23 path expressions [15]. Improvements of this sort are possible for any SI-only query, because cloaking dynamically creates a view that hides non-SI nodes.

We now share some experimental results illustrating improvement in query-execution efficiency. Table 1 shows the four datasets used in the experiments. All marks in these datasets were to PDF documents which represented conference papers. The performance of the bi-level query processor was compared to the traditional query processor included in the .NET Framework. In all, we executed 219 queries (including queries over other datasets), but, for brevity, we report the result of only two queries. We executed each query thrice and report the average execution time.

Table 1. SISRS datasets used in experimental evaluation of the bi-level query processor

			Number of mark associations		
Document	**File size (KB)**	**# Base docs**	**EMark**	**AMark**	**Total**
SISRS-1	206	53	1,908	53	1,961
SISRS-2	414	106	3,816	106	3,922
SISRS-4	833	213	7,668	213	7,881
SISRS-8	1593	426	15,336	426	15,762

SI-only queries: We first report on the performance of the bi-level query processor on the query to retrieve all SI from SISRS documents. Again, with the bi-level query processor, the simple white XPath expression "." was used, whereas an XSLT style sheet was used with the traditional processor. Table 2 compares the performance of the two processors for this task and shows that cloaking saved 50% or more time when retrieving SI.

When retrieving SI, the bi-level processor is faster for two reasons: First, it examines fewer nodes (62% fewer node movements than the traditional processor). Second, the use of XPath allows existing nodes to be returned as they are, whereas XSLT always constructs new nodes. Thus, the bi-level processor also consumes less memory when executing this query.

Table 2. Time (in milliseconds) to retrieve SI only using the bi-level query processor and the traditional query processor

Processor (language)	SISRS-1	SISRS-2	SISRS-4	SISRS-8
Bi-level (XPath)	5.21	10.42	20.83	36.46
Traditional (XSLT)	10.42	20.83	41.67	78.12

Focused and un-focused path expressions: Cloaking encourages the use of unfocused path expressions where a user might otherwise use focused expressions. A *focused path expression* is an expression that guides the query processor strictly along the path of interest. An *unfocused path expression* does not guide the processor in this manner. A focused expression causes fewer navigator movements, but is sensitive to schema revisions. Unfocused expressions frequently employ wildcards and other shortcuts that ease query development, but they increase navigator movements. Our observation is that cloaking reduces the number of navigator movements for unfocused expressions, thereby reducing query-execution effort while making it easier to express queries, a quality especially useful in ad-hoc querying. (Some schemas might provide greater benefits than others.)

Table 3 compares the time to evaluate a focused expression and its equivalent unfocused expression in both the bi-level processor and the traditional processor. Whereas the bi-level processor provides the obvious savings over the traditional processor in each case, it also considerably reduces the overhead of using unfocused expressions: 25% overhead (9705 ms instead of 7742 ms) with the bi-level processor compared to 75% with the traditional processor.

Table 3. Time (in milliseconds) to retrieve comment text using focused and unfocused path expressions

Path expression	Bi-level	Traditional	Savings due to Bi-level
Focused: `/Reviews/Paper/Comment/text()`	5834	7742	33%
Unfocused: `//text()`	9705	13521	39%

5 Related Work

Today, XML-management largely takes place in two settings: relational database management systems (RDBMSs) and standalone XML processors. The exact techniques commercial RDBMSs use for XML management are generally unpublished, but we expect they shred XML data into relations and then employ traditional relational techniques for querying. Systems such as Microsoft SQL Server [4] also allow creation of XML views of relational data.

Standalone XML processors generally specialize in XML querying (not in XML storage). Conceptually, our approach is closer to that of a stand-alone processor (our bi-level query processor is implemented as one), but nothing in our approach should prevent DBMSs from employing cloaking in their query processing.

As outlined in Section 1, using indexes is a way to improve query-execution efficiency. Luk and others [12] discuss the use of path-based indexes and other kinds of indexes in XML querying, but they do not explicitly consider the problem of accessing a systematic subset of an XML document as we do with cloaking.

There has been interest in XML views for over a decade. For example, Abiteboul [1] discusses various aspects involved in supporting XML views including the benefits of assigning an object identifier to each subdocument so that you can refer to it even if the path changes. In a similar way, in our model, we explicitly identify nodes so that we can attach additional information such as the colors that have been assigned.

The ActiveViews framework [2] introduces a feature that is in some sense a dual to cloaking. The "with" clause in a view definition can specify what portions of a document or element are retained in a view; cloaking, in contrast, specifies portions of a document that are to be suppressed. Also, the "with" clause appears to operate only at the schema level; our approach can operate at both the schema and instance levels.

Perhaps the work most related to cloaking involves XML views expressed using XSLT [8] where queries against a view are translated based on the XSLT view definition into queries addressing the underlying XML documents. Query answers are then translated back into the view, also based on the XSLT view definition. This approach, as well as our approach using cloaking, refrains from transforming and materializing the underlying XML documents based on the view definition. In their work, they modify the (XPath) query appropriately whereas with cloaking we modify the navigation operator that traverses the XML document, effectively "lying" to the XML path processor when nodes have been cloaked.

Both view-materialization and view-unfolding are related to cloaking. View materialization aids repeated use of views and is preferred when the view is used without refinement (such as filtering out parts of the generated view). View unfolding does not materialize results and is beneficial when views are refined before use. View materialization is popular in data warehouses, whereas view unfolding is employed in data-security and data-privacy systems (for example, SMOQE [6] and Hippocratic databases [11]).

Cloaking can be integrated into view unfolding, because, as outlined in Section 3.2, our test functions are just conjunctive terms. Specifically in the XML model, cloaking aids the use of non-materialized views, but we see benefits to cloaking even when a view is materialized, because view materialization *follows* view generation and cloaking relates to view generation. (To materialize a view, the view has to be first generated. The query processor can employ cloaking to generate the view.)

Data-privacy systems effectively cloak data, but cloaking in that context is an end, not a means. Specifically, published literature does not show privacy systems using cloaking to improve view creation, query expression, or query execution.

The compatibility issues of our cloaking model with the relational model (highlighted in Section 3.3) exist in data-privacy systems as well. Privacy systems typically manage the issues by disallowing queries that address hidden elements and impose structural constraints such as container visibility. Similar model-specific constraints may be imposed in an application of our model as well.

Due to the use of colors, our cloaking approach appears similar to the *multi-colored tree model* (MCT) [9], but MCT is not designed for cloaking. (Instead, it aims to simplify query expression over shallow trees that result from normalization of nested data.) In MCT, each XML document tree has a color. An element may be used in multiple document trees, and each element is implicitly assigned the set of colors formed by collecting the color of each tree in which the element is used. Attributes, namespaces, and non-element child nodes are assigned the same colors as the parent element. In our approach colors are assigned to tree nodes, not to trees; and *any kind of node* may be colored without constraints.

MCT requires the use of *MCXQuery*, an extension of XQuery. MCXQuery allows each step in a path expression to choose the tree in which navigation continues by including a color in each step. Our approach uses existing query languages as they are, and a node's color determines if a node is visible to a query; not navigation paths.

Buneman and others [3] use colors to represent data provenance in a nested relational model. Like us, they allow a different color to be assigned to tuples, fields, and to entire relations, but they allow only one color per "object" and they introduce a new primitive data type called "color". Also, they require the use of nested relational algebra extended with "provenance aware" semantics for querying.

MONDRIAN [7], a system to represent and propagate annotations, uses an extension of the relational model to represent annotations associated with *colored blocks*, and an extension of relational algebra, called *color algebra*, to propagate annotations. Colors are associated with operators within a query. Only three operators employ colors, of which only one operator accepts a color. The other two operators have fixed semantics to propagate colors, in effect fixing a "cloaking scheme". In contrast, our cloaking schemes are not fixed and the schemes are independent of operators and queries. (The same cloaking scheme may be used for different queries.) Also, in our approach, a color is assigned to an entire query, not to individual operators of a query.

6 Summary

We have introduced the notion of *cloaking*, a means of hiding data parts and selectively revealing cloaked parts only to qualifying queries. We have presented both a formal model and an architectural reference model (which are independent of applications and data models) of a cloaking query processor and showed how a query processor can implement cloaking without altering its input data. We have also illustrated how cloaking improves view creation, query expression, and query execution in both bi-level query and non-bi-level query settings.

Acknowledgments. This work was supported in part by NSF project numbers 0511050 and 0534762. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Abiteboul, S.: On Views and XML. In: Proceedings of 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 1999), Philadephia, PA (1999)

2. Abiteboul, S., Amann, B., Cluet, S., Eyal, A., Mignet, L., Milo, T.: Active Views for Electronic Commerce. In: Proceedings of 25th International Conference on Very Large Databases (VLDB 1999), Edinburgh, Scotland (1999)
3. Buneman, P., Cheney, J., Vansummeren, S.: On the Expressiveness of Implicit Provenance in Query and Update Languages. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 209–223. Springer, Heidelberg (2006)
4. Creating XML Views by Using Annotated XSD Schemas. Microsoft Corporation, `http://msdn.microsoft.com/en-us/library/aa258637(v=sql.80).aspx` (accessed October 10, 2011)
5. Delcambre, L., Maier, D., Bowers, S., Weaver, M., Deng, L., Gorman, P., Ash, J., Lavelle, M., Lyman, J.: Bundles in Captivity: An Application of Superimposed Information. In: Proceedings of ICDE 2001, Heidelberg, Germany (2001)
6. Fan, W., Geerts, F., Jia, X., Kementsietsidis, A.: SMOQE: A System for Providing Secure Access to XML. In: Proceedings of 32nd International Conference on Very Large Data Bases (VLDB 2006), Seoul, Korea, September 12-16 (2006)
7. Geerts, F., Kementsietsidis, A., Milano, D.: MONDRIAN: Annotating and Querying Databases through Colors and Blocks. In: Proceedings of 22nd International Conference on Data Engineering (ICDE 2006), Atlanta, GA, April 3-7 (2006)
8. Groppe, S., Bottcher, S., Birkenheuer, G., Hoing, A.: Reformulating XPath Queries and XSLT Queries on XSLT Views. Data & Knowledge Engineering 57(1), 64–110 (2006)
9. Jagadish, H.V., Lakshmanan, L.V.S., Scannapieco, M., Srivastava, D., Wiwatwattana, N.: Colorful XML: One Hierarchy Isn't Enough. In: Proceedings of SIGMOD 2004, Paris, France (2004)
10. Kay, M.H.: SAXON: The XSLT and XQuery Processor 8.8, `http://saxon.sourceforge.net/` (accessed October 10, 2011)
11. LeFevre, K., Agrawal, R., Ercegovac, V., Ramakrishnan, R., Xu, Y., DeWitt, D.J.: Limiting Disclosure in Hippocratic Databases. In: Proceedings of 30th International Conference on Very Large Data Bases (VLDB 2004), Toronto, Canada, August 31-September 3 (2004)
12. Luk, R.W.P., Leong, H.V., Dillon, T.S., Chan, A.T.S., Croft, W.B., Allan, J.: A Survey in Indexing and Searching XML Documents. Journal of the American Society for Information Science and Technology 53(6), 415–437 (2002)
13. Maier, D., Delcambre, L.: Superimposed Information for the Internet. In: Proceedings of WebDB 1999, Philadelphia, PA (1999)
14. Murthy, S., Maier, D.: SISRS: The Superimposed Scholarly Review System (2004), `http://sparce.cs.pdx.edu/pubs/SISRS-WP.pdf` (accessed October 10, 2011)
15. Murthy, S.: A Framework for Superimposed Applications: Techniques to Represent, Access, Transform, and Interchange Bi-level Information, Computer Science, Portland State University (2009)
16. Murthy, S., Maier, D., Delcambre, L., Bowers, S.: Putting Integrated Information in Context: Superimposing Conceptual Models with SPARCE. In: Proceedings of First Asia-Pacific Conference of Conceptual Modeling, Dunedin, New Zealand, January 22 (2004)
17. .NET Framework Developer Center. Microsoft Corporation, `http://msdn.microsoft.com/netframework/` (accessed October 10, 2011)
18. Sixml. The Else Institute, `http://www.sixml.org` (accessed October 10, 2011)
19. XML Path Language (XPath) Version 1.0 (1999) W3C, `http://www.w3.org/TR/xpath` (accessed October 10, 2011)
20. XQuery 1.0 and XPath 2.0 Data Model (XDM) (2007) W3C, `http://www.w3.org/TR/xpath-datamodel/` (accessed October 10, 2011)

On Models of Concepts and Data

Arne Sølvberg

NTNU – Norwegian University of Science and Technology, Trondheim, Norway

Abstract. Bernhard Thalheim's landmark book titled "Entity-Relationship Modeling: Foundations of Database Technology" [9] starts with listings of weaknesses and strengths of contemporary approaches to database models. His analysis is as valid today as when it was published in 2000. An important feature of Information Systems Engineering is the need to master software and data systems complexity while at the same time being able to present design details in a comprehensible way. The trend towards embedded systems necessitates that software and data models must be integrated with models of the application domains, be they technical, organizational, people centered or a mixture thereof. What follows is a discussion of possible underlying causes of the weaknesses as stated by Bernhard Thalheim, as well as proposals to approaches for removing the weaknesses.

1 Introduction

In the introductory chapter of his book [9] Bernhard Thalheim states that "The problem of database design can be stated as follows:

Design the logical and physical structure of a database in a given database management system (or for a database paradigm) so that it contains all the information required by the user and for the efficient behavior of the whole information system for all users and with application processes and user interaction.
"

This indicates that the whole information system serves as reference for validating a database design. So the modeling concepts of the data model must be related to the modeling concepts of the information system model. Furthermore, an information system is usually built in order to support the operation of some other system which we will call the usage domain, e.g., by keeping track of its state-of-affairs and making this information available to actors of "the other system". This "usage domain" is known by many different names, e.g., the user domain, the business processes, the application domain, the Universe of Discourse (UoD), "the real world". The usage domain serves as reference for the validation of the information system. Consequently the modeling concepts of the two must be related in some way or another. So we end up by requiring that the modeling concepts of the data base and software models, the information system models and the usage domain models be related in some transparent way for a proper design evaluation to take place.

The requirement of model integration cannot be satisfied with contemporary modeling approaches. The practical solution to the dilemma is to treat separately the modeling in each of the three realms: the systems of the usage domain are modeled with usage domain modeling languages which reflect on usage domain theories, the data and software systems are modeled with programming languages and data modeling languages, and the information systems are modeled with modeling languages for this domain.

The transformations between the three modeling domains are mostly described informally, either as textual statements alone, or in modeling languages created for such purposes. An example of the latter is “behavior modeling”. The information system is here seen as a structure of information stores and associated information processes which relates to the usage domain via event notices that are interchanged between the software/data domain and the usage domain. User interface design and other types of protocol designs for exchanging information between the two are associated with behavior modeling.

Information systems are closely related to people oriented usage domains. For usage domains where no people are involved the interchange protocols between the usage domain and the data/software domain can be designed directly without much consideration to the comprehension aspects of data. “Comprehension” is hardcoded in the interchange protocols. Only when people participate as actors in information processes, comprehension becomes an issue. Those who interact with a data system must understand what the data of their transactions means.

There is no generally agreed system model for persons seen as knowledge processors. For the brake system of a car there are models, as there are for the manufacturing of steel and for the cost optimal transportation of goods. Information systems can be seen as models for the behavior of persons (and organizations of persons) in their roles as knowledge consumers and knowledge producers interacting with data/software systems. So the information systems domain may well be regarded as a usage domain in its own right. It is quite common that information systems interact with systems in other usage domains. So it becomes practical to classify information systems as belonging to a separate domain in their own right. Increasingly whole systems consist of interacting subsystems of software/data, people and “other systems”.

The modeling regime of the information systems domain is “weaker” than for the two other domains. The modeling regime of the data/software domain reflects the technological practices of more than half-a-century of building computers, databases and software. The modeling regimes of the physical world reflect on the laws of nature as well as the practices of building technical products of various kinds. Systems designers who do not respect those laws and practices will not be successful. The information systems domain is different. There are no “laws” to obey or to use for guidance and there are no agreed models of human behavior in information processing situations.

Many of the unsolved challenges in application systems design appear as weaknesses in information systems modeling. Examples are semantic modeling, lacking comprehension of systems specifications, lacking integration of systems models, and so on. The remainder of this paper contains a discussion of some of the central (in the author’s opinion) aspects of the information system modeling regime. The paper may be seen as an extension of a recent discussion in a similar paper of 2010 [8].

2 Contemporary Data Modeling – The ER Model

Bernhard Thalheim gives in [9] a comprehensive review of contemporary data modeling theory and practice, see [9] chapters 1-3. In [9] chapter 4, he introduces the HERM extension to the Entity-Relationship Model, in order to overcome some of the stated weaknesses in the original ER-model [2]. Even if the HERM extension represents a big step forward it still suffers from some of the inherent weaknesses that the ER-model shares with other modeling approaches, e.g., with the relational model of data.

The goal of data base modeling is to design an efficient and appropriate data base. Some of the most important criteria are performance, integrity, comprehensibility and expandability, see [9] p9. Each of these put different requirements on the data modeling language. The performance criterion requires that the data model must reflect the essential properties of data storage and data processing by computers. The integrity criterion requires that the data model must reflect the rules of the usage domain. Comprehensibility means that the data model must be understandable for human beings, and expandability means that new versions of the data model both must reflect the changes in the usage domain, and relate to already existing databases that refer to previous versions of the data model.

No existing data modeling language satisfies these conflicting requirements. The most important criteria to satisfy have been performance and integrity. The chosen modeling concepts reflect this. Comprehension and expandability require modeling concepts for capturing the essentials of the usage domains, including the people domain (the information system domain). The Entity-Relationship model goes some way in satisfying the comprehension and expandability criteria, at the expense of satisfying the performance criterion.

The original intention for the ER-model was to provide an approach to modeling in the usage domain. Judging from the examples of applying the ER-model one has to conclude that the modeling concepts are mostly used for the modeling of data instead of the modeling of phenomena in the usage domain. This is reflected in the many definitions of the entity-concept, see [9] p4. The various (informal) definitions range from viewing “an entity is a thing which can be distinctly identified”, via “an entity is a specific thing about which information is kept”, to “the term entity refers to the logical representation of data”. One may safely conclude that the practical realities of the performance criterion being “the elephant in the room” have overtaken the noble intentions of representing realistic models of usage domains. However, the popularity of the ER-model and its extensions proves that in spite of this it still serves to some degree to satisfy the criterion of comprehensibility.

Two interesting aspects of the data model problem are the relation between data and usage domain phenomena, and the model characteristics of usage domains. One interesting observation is that the contemporary approaches to data modeling are solidly based on a set theoretic view of the world. Classical technology is, nevertheless, based on parthood semantics: mereology. This will be discussed in the following.

3 Mismatch between Modeling in the Data and Usage Domains

A data base consists of values which represent objects in the usage domain. Data modeling languages must be adequate for the purpose of designing databases with satisfying performance. Bernhard Thalheim introduces three types of adequacy, see [9] p14:

- *Metaphysical adequacy* is achieved if there are no contradictions between the objects we wish to represent and our representation of them
- *Epistemological adequacy* regards the ability to express things in the real world. To be epistemological adequate the representation must be capable of being used to express the things that we know about the real world
- *Heuristic adequacy* is achieved if we can express database processing in our representation.

Data semantics deal with the explicit relationships between the data in the database and the real world phenomena that are represented by the data. This relationship is straight forward if the data model is metaphysical adequate. The ER model is metaphysical adequate for usage domains where all objects are discrete and can be described in terms of set theory. But this is generally not the case. Most theories of usage domains are not set-oriented at all. On the contrary, most world models reflect a world of continuous phenomena. Discontinuities are very rare in nature.

An illustrative example is the modeling of copper. A set theoretic (ER-type) modeling would have to assume that the extension of the concept of copper is all of the copper metal in the world, including all things made of copper, and including all of the copper metal that is found in alloys and ores. Furthermore, each copper sample is itself copper, so each sample is therefore part of itself. Set oriented modeling is not suitable for this situation.

The deficiencies of set theory may be counteracted by introducing mereology – the theory of parthood relations - as a mathematical foundation of similar importance as set theory, see [13], and [8]. This makes it possible within a respectable mathematical framework, to express the alternative (but still similar) classification of facts which is found in the American National Standard's guidelines for thesauri construction [11]. The standard document recommends that distinction is made among the following kinds of concepts (non-exhaustive list):

- things and their physical parts, e.g. bird, car, mountain
- materials, e.g., water, steel, oxygen
- activities or processes, e.g., painting, golf
- events or occurrences, e.g., birthday, war, revolution
- properties or states of persons, things, materials or actions, e.g., elasticity, speed
- disciplines or subject fields, e.g., theology, informatics
- units of measurement, e.g., hertz, volt, meter.

We must conclude that the ER model is not epistemological adequate. We are not able to express what we know about the physical world in the ER model. In practice we find ways around this deficiency by enforcing a set theoretic view on the world, and hiding the epistemological mismatch in heuristics. An excellent discussion of the challenges involved is offered by Bernhard Thalheim in [10].

The mismatch nevertheless creates difficulties in expressing the semantics of the ER model of data and of associated databases. A fundamental challenge is to provide a theory that relates the different modeling regimes, e.g., mereology and set theory. It may prove difficult to provide ER-extensions to fix the mismatch problem unless the theoretical basis is in order.

4 Modeling in the People Domain: The Concept of Reflexivity

More than 30 years ago Lehman introduced a classification of software in S-, P-, and E-programs [4]. S- and P-programs are seen as being separate from their usage domains, while E-programs are seen as being embedded in their usage domains. An E-program is itself part of the universe that it both models and controls. The E-program contains a model of its own interaction with its operational environment. The E-program designer must perceive the combined behavior of the E-program and its environment as it will be once the program has been implemented and installed. The E-programs are true evolutionary programs.

E-programs may be seen as particularizations of the more general concept of reflexivity [12], which refers to circular relationships between cause and effect. In our context reflexivity means that the model of a usage domain influences the individuals' perception of the domain. So when the model changes so does the perception. In this way the model becomes part of the domain. The financier George Soros claims that an understanding of the principle of reflexivity is the secret behind his success as an investor [7]. Reflexivity is important for modeling in the people domain, that is, it is important for information systems engineering.

The essential feature of the principle of reflexivity applied to information systems engineering is that every new information process, with associated data and computer programs, becomes parts of the usage domain and consequently modifies the model of the usage domain. Whenever people participate in the usage domain, the principle of reflexivity guarantees that information systems will evolve with time. But the principle of reflexivity also guarantees that it is impossible to develop a final model of a people oriented usage domain. Every act of modeling that we do changes the usage domain.

Data semantics reflect relationships between the contents of a database and the model of the world that is represented by the data. The continuous evolution of the domain models also means that the modeling concepts evolve. In order to keep track of the evolution of the semantics of data it becomes necessary to keep track of the evolution of the modeling concepts. New concepts must relate to old concepts in order for people to understand the essentials of new models. This must be taken into consideration when designing approaches for capturing data semantics.

5 Concept and Data

We shall have a closer look at the relationship between data and concept. We follow the usual distinction among concepts, terms and referents [6]. Terms are linguistic units, and concepts are units of thought. Referents are what the terms and concepts refer to in the “real world”. Linguistic units are represented by digital numbers, that is, by data. Concepts are denoted by terms. A name is a term. Labels are unique terms within a name space. Concepts are uniquely identified within a name space if they are denoted by labels from that namespace.

Bunge classifies concepts of natural sciences in four groups: individual concepts, class concepts, relational concepts and magnitudes (quantitative concepts) [1] . Individual concepts refer to individuals, e.g., the individual concept “Obama” refers to a particular individual. Class concepts refer to classes of individuals sharing some similarities, e.g., “US presidents”. Relational concepts refer to relationships between class concepts. Distinctions are made between specific (definite) concepts and generic (indefinite) concepts, e.g., “Obama” is a specific concept, but “x” is a generic concept and denotes an arbitrary referent.

Quantitative concepts apply to properties that reflect magnitudes associated with individuals and/or sets, e.g., the temperature of a body, the number of elements of a set. No distinct object is associated with a quantitative concept. Functions are the structure of quantitative concepts, e.g., weight, mass, heat, acceleration. For example, weight is a function that maps the set of bodies (each of which has a weight) into the set of real numbers. Quantitative concepts are thus the conceptual core of measurement [1]. Quantitative concepts relate individual, class and relational concepts to values, that is, to data.

Models are conceptual systems that attempt to represent some interrelated aspects of real systems [1]. Quantitative concepts represent the essential details of a model because they permit the representation of magnitudes, e.g., the weight of a person. Models for some usage domain can be designed even if there is no available theory for the domain. Theories are systems of hypothesis about how referents in a usage domain as represented by relevant concepts are interrelated. E.g., an integrity constraint in a database schema may be viewed as an expression of a part of a theory. Models are expressed by data which represent the labels of the involved concepts as well as the values of the involved quantitative concepts.

The Entity-Relationship model of data has the three modeling concepts entity type, relationship type and attribute, see [9] and [2]. In our context attributes are data types that represent quantitative concepts and their values, as well as the scale and the precision of the values [8]. Assume that we have defined the class concept PERSON and the quantitative concepts WEIGHT and P-ID, where P-ID is a 1:1 function that maps from PERSON to a value set of unique labels for individuals in the extension of PERSON. A valid expression for this situation in the ER-model of data could be ERperson(ERpid, ERweight) where ERperson is an entity type, ERpid is a key attribute and ERweight is a non-key attribute. Then ERpid and ERweight are data types with a stated precision and a stated scale, e.g., ERweight may be measured in kilograms with a precision of three decimals.

Every physical body has a weight. There are many scales for measuring weights. The level of precision depends on the purpose of recording weights. Class concepts are established to serve the purpose of the model designer. So are attributes. E.g., the weights of Americans are measured in pounds, the weights of Europeans are measured in kilograms. When establishing entity types for the two situations we would probably get something like ERamerican(SSN, ERamericanweight) and EReuropean(ER-some-id, EReuropeanweight). The two weight attributes would probably be of different scales and different precisions. In the ER-model there is no explicit way of establishing that the two weight-attributes refer to the same quantitative concept of WEIGHT.

Furthermore, WEIGHT is a concept whose measurement values depend on the location of the measurement. The higher the altitude, the less is the weight of a given mass. This is of interest if the required precision is high, otherwise the altitude effect can be disregarded.

6 Conclusion

Prospects for further progress in data semantic modeling may be bleak if we do not bring quantitative concepts more directly into data modeling. The ER model seems to be a good candidate if properly extended. Possible extension should include appropriate relations between attributes and quantitative concepts.

In practice it seems that the applications of the ER model assume metaphysical adequacy, that is, there are no contradictions between the objects we wish to represent and our representation of them. So, while an entity type in theory is assumed to represent referents in "the real world", in practice the entity types represent data about the referents. A consequence of this is that the inherent structure of data bases has to be superimposed on the models of the world, in order to achieve metaphysical adequacy. This limitation should be removed in order to achieve better descriptions of usage domains and their associated data semantics. A possible way forward may be to bring parthood semantics into the modeling loop, and not only rely on set semantics.

References

1. Bunge, M.: The Philosophy of Science. Transaction Publishers (1998) ISBN 0-7658-0415-8
2. Chen, P.P.: The Entity-Relationship Model: Toward a unified view of data. ACM TODS 1(1), 9–36 (1976)
3. Krogstie, J., Opdahl, A.L., Brinkkemper, S. (eds.): Conceptual Modeling in Information Systems Engineering. Springer, Heidelberg (2007) ISBN 978-3-540-72676-0
4. Lehman, M.M.: Programs, life cycles, and laws of software evolution. Proceedings of the IEEE, 1060–1076 (September 1980)
5. Nurcan, S., Salinesi, C., Souveyet, C., Ralyté, J. (eds.): Intentional Perspectives on Information Systems Engineering. Springer, Heidelberg (2010) ISBN 978-3-642-12543-0
6. Ogden, C.K., Richards, I.A. (eds.): The meaning of meaning. Kegan, Paul, Trench, Trubner, London (1923)
7. Soros, G.: The Alchemy of Finance. Simon & Schuster (1988); ISBN 0-671-66338-4, paperback: pp 49–51. Wiley (2003) ISBN 0-471-44549-5)
8. Sølvberg, A.: On Roles of Models in Information Systems. In: [5], pp. 17–38
9. Thalheim, B.: Entity-Relationship Modeling: Foundations of Database Technology. Springer, Heidelberg (2000) ISBN 3-540-65470-4
10. Thalheim, B.: Challenges to Conceptual Modeling. In: [3], pp. 58–74
11. ANSI/NISO Z39.19-1993: Guidelines for the Construction, Format, and Management of Monolingual Thesauri, ISBN 1-880124-04-1
12. `http://en.wikipedia.org/wiki/Reflexivity_social_theory` (accessed July 15, 2011)
13. `http://plato.stanford.edu/entries/mereology/` (accessed July 15, 2011)

Model Transformation By-Example: A Survey of the First Wave*

Gerti Kappel[1], Philip Langer[2], Werner Retschitzegger[2], Wieland Schwinger[2], and Manuel Wimmer[1]

[1] Vienna University of Technology, Austria
{gerti,wimmer}@big.tuwien.ac.at
[2] Johannes Kepler University Linz, Austria
{langer,retschitzegger,schwinger}@jku.at

Abstract. Model-Driven Engineering (MDE) places models as first-class artifacts throughout the software lifecycle. In this context, model transformations are crucial for the success of MDE, being comparable in role and importance to compilers for high-level programming languages. Thus, several model transformation approaches have been developed in the last decade, whereby originally most of them are based on the abstract syntax of modeling languages. However, this implementation specific focus makes it difficult for modelers to develop model transformations, because they are familiar with the concrete syntax but not with its computer internal representation.

To tackle this problem, model transformation by-example approaches have been proposed which follow the same fundamental idea as query by-example and programming by-example approaches. Instead of using the computer internal representation of models, examples represented in concrete syntax are used to define transformations. Because different transformation scenarios occur in MDE, different by-example approaches have been developed. This chapter gives an overview on the emerging concepts, techniques, and approaches in this young by-example area.

Keywords: model transformation, by-example, model-driven engineering.

1 Introduction

Model-Driven Engineering (MDE) places models as first-class artifacts throughout the software lifecycle [7,16,38]. In this context, model transformations [39] are crucial for the success of MDE, being comparable in role and importance to compilers for high-level programming languages, for bridging the gap between design and implementation. Thus, several model transformation approaches (cf. [13] for an overview) have been developed in the last decade, whereas most of them are based on the abstract syntax of modeling languages which is defined by

* This work has been partly funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) and FFG under grant FIT-IT-819584.

so-called metamodels [26]. Metamodels describe by a limited set of UML class diagram concepts the object structure for computer internally representing and persisting models. However, this implementation specific focus makes it difficult for modelers to develop model transformations, because modelers are mainly familiar with the concrete syntax of the modeling languages (i.e., their notation) and not with their metamodels. This is aggravated by the fact that metamodels may become very large: for instance, the UML 2 metamodel [34] has about 260 metamodel classes [30]. Moreover, some language concepts, which have a particular representation in the concrete syntax, are not even explicitly represented in the metamodel. Instead, these concepts are hidden in the metamodel and may only be derived by using specific combinations of attribute values and links between objects [20]. Thus, they are often hard to discover as illustrated in the example in Fig. 1.

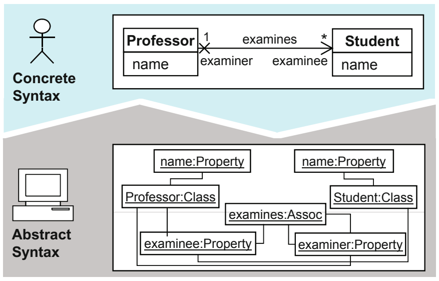


Fig. 1. Gap between user intention and computer internal representation

To tackle this problem, model transformation by-example (MTBE) approaches [45,47] have been proposed which follow the same fundamental idea as *query by-example* developed for querying database systems by giving examples of query results [48] and *programming by-example* for demonstrating actions which are recorded as replayable macros [29]. This means, instead of using the computer internal representation of models, MTBE allows to define transformations using examples represented in concrete syntax. Consequently, the modeler's knowledge about the notation of the modeling language is sufficient.

Because different transformation scenarios occur in MDE [31], different MTBE approaches have been developed in the last years. In general, two kinds of approaches may be distinguished: (i) Approaches following a *demonstration-based* approach, meaning that model transformations are demonstrated in the modeling editor by modifying example models. These modifications are recorded and from the concrete changes, the general transformation is derived which may be replayed on other models as well. (ii) Approaches which follow a *correspondence-based* approach. Instead of demonstrating the transformation in modeling editors, the input model, the output model as well as the correspondences between them have to be given by the user. For both kinds, a multitude of approaches have been proposed during the last years [4,8,14,17,23,27,43,45,47].

2 MDE in a Nutshell

Before MTBE is presented, the prerequisites, i.e., the core techniques of MDE, are explained. First, the essence of modeling language engineering is outlined to illustrate how models are represented in the context of MDE, and subsequently, the main principles and patterns of model transformations are introduced.

2.1 Modeling Language Engineering

Modeling language engineering in MDE comprises at least two components [24]. First, the abstract syntax of a language has to be defined by a metamodel, i.e., a model defining the grammar of the language. Second, to make a language more usable, a mapping of abstract syntax elements to concrete syntax elements (such as rectangles, edges, and labels) has to be provided. In the following, an example-based description of defining a modeling language is given.

Abstract Syntax. Similar as EBNF-based grammars [18] for programming languages, metamodels represent the concepts and their interrelationships of modeling languages. The most widely used formalism to define metamodels in MDE is the Meta Object Facility [33] (MOF), which is a standardized language to define modeling languages based on the core concepts of UML class diagrams (*classes*, *attributes*, and *references*). In the upper part of Fig. 2, an excerpt of the kernel of the UML metamodel is represented in terms of MOF concepts.

The aim of metamodeling lies primarily in defining modeling languages in an object-oriented manner leading also to efficient repository implementations for storing and retrieving models. This means that in a metamodel not necessarily all modeling concepts are represented as first-class citizens. Instead, the concepts are frequently hidden in attributes and in references. We call this phenomenon *concept hiding* (cf. [20] for an in-depth discussion).

By instantiating metamodels, models are created. An instantiation is represented by a UML object diagram comprising of *objects* as instances of *classes*, *values* as instances of *attributes*, and *links* as instances of *references* as e.g., depicted in the middle part of Fig. 2. It has to be noted that in contrast to EBNF-based grammars, metamodels do not define the concrete syntax of the languages. Thus, only generic object graphs as depicted in the middle part of Fig. 2 may be created. The concrete syntax has to be defined in addition to the metamodel which is explained next.

Concrete Syntax. The concrete syntax of modeling languages [2] comprises in most cases graphical elements such as ellipse, label, and rectangle, which may be further combined to more complex forms. The actual notation of modeling concepts is defined by a mapping of abstract syntax elements to concrete syntax elements. The mappings may be expressed in triples of the following form:

$$Triple := < as\_E, cs\_E, const(as\_E)? > \quad (1)$$

The first part *as_E* stands for an element of the *abstract syntax*, the second *cs_E* for an element of the *concrete syntax*, and the last *const(as_E)* stands

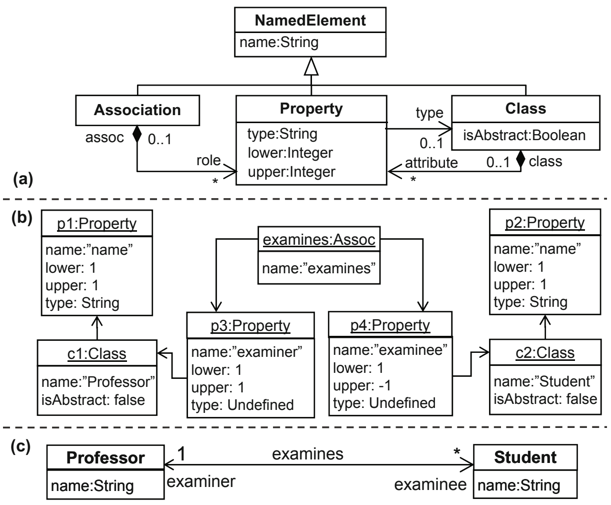


Fig. 2. (Meta)modeling: (a) UML Metamodel Kernel, (b) Example Model in AS, and (c) Example Model in CS

for an optional constraint, mostly defined in the Object Constraint Language (OCL) [36], that defines under which conditions, i.e., links and attribute values of an *as_E* element, this element is represented by a *cs_E* element. In case no constraint is defined, there is a *one-to-one* mapping between an abstract syntax element and a concrete syntax element, i.e., the concept defined in the metamodel is directly represented by one concrete notational element. However, the other case is the more interesting one in the context of MTBE. The presence of a constraint defines a new concept for the notation layer, which is not explicitly represented by one of the metamodel classes. Consequently, when defining model transformations based on the abstract syntax, the constraints for these concepts must be defined by the user. This is a tedious and error-prone task that requires excellent knowledge about the metamodel.

When considering our running example, for instance, the `Class` concept is mapped to `Rectangle` and `Class.name` is mapped to `Label`. By using such mappings, the UML object diagram shown in the middle part of Fig. 2 may be rendered as shown in the lower part of Fig. 2 by graphical modeling editors.

2.2 Transformation Engineering

In general, a model transformation takes a model as input and generates a model as output[1]. Mens et al. [31] distinguish between two kinds of model transformations: (i) *exogenous transformations* a.k.a. model-to-model transformations

[1] Also several input models and output models may be possible, but in the scope of this paper, such settings are not considered.

or out-place transformations, in which the source and target metamodels are *distinct*, e.g., transforming UML class diagrams to relational models, and (ii) *endogenous transformations* a.k.a. in-place transformations, in which the source and target metamodels are the *same*, e.g., a refactoring of a UML class diagram. In the following, we elaborate on these two kinds in more detail.

Exogenous Transformations. Exogenous transformations are used both to exploit the constructive nature of models in terms of *vertical transformations*, thereby changing the level of abstraction and building the bases for code generation, and for *horizontal transformation* of models that are at the same level of abstraction [31]. Horizontal transformations are of specific interest to realize different integration scenarios, e.g., translating a UML class model into an Entity Relationship (ER) model. In vertical and horizontal exogenous transformations, the complete output model has to be built from scratch.

Endogenous Transformations. In contrast to exogenous transformations, endogenous transformation only rewrite the input model to produce the output model. For this, the first step is the identification of model elements to rewrite, and in the second step these elements are updated, added, and deleted. Endogenous transformations are applied for different tasks such as model refactoring, optimization, evolution, and simulation, to name just a few.

Model Transformation Languages. Various model transformation approaches have been proposed in the past decade, mostly based on either a mixture of declarative and imperative concepts, such as ATL [19], ETL [25], and RubyTL [12], or on graph transformations, such as AGG [44] and Fujaba [32], or on relations, such as MTF[2] and TGG [1]. Moreover, the Object Management Group (OMG) has published the model transformation standard QVT [35] which is currently only partly adopted by industry. Summarizing, all approaches describe model transformations by rules using metamodel elements, whereas the rules are executed on the model layer for transforming a source model into a target model. Rules comprise *in-patterns* and *out-patterns*. The in-pattern defines when a rule is actually applicable as well as retrieves the necessary model elements for computing the result of a rule by querying the input model. The out-pattern describes what the effect of a rule is, such as which elements are created, updated, and deleted. All mentioned approaches are based on the abstract syntax of modeling languages only, and the notation of the modeling language is totally neglected.

Defining model transformations by using the abstract syntax of modeling languages comes on the one hand with the benefit of the generic applicability. On the other hand, the creation of such transformations is often complicated and their readability is much lower compared to working with the concrete syntax [3,28,41,45,46]. Therefore, MTBE approaches have been proposed to use the concrete syntax of modeling languages for defining model transformations. In the following two sections, we present the essence of MTBE for endogenous transformations as well as for exogenous transformations.

[2] `http://www.alphaworks.ibm.com/tech/mtf`

3 MTBE for Endogenous Transformations

For endogenous transformations, two dedicated by-example approaches [8,43] have been proposed in the last years that can be seen as a special kind of MTBE called *Model Transformation By Demonstration* (MTBD). MTBD exploits the *edit operations* demonstrated on an example model in order to obtain transformation specifications that are also applicable to other models. Interestingly, no *correspondence-based* approach has been proposed for endogenous transformations, so far. In the following, we present the common process of both MTBD approaches for specifying an endogenous transformation by demonstration and show how this process is applied to a concrete model refactoring example. Finally, we conclude this section by elaborating on the peculiarities of both MTBD approaches.

3.1 Process

In general, the MTBD process consists of two phases: (1) demonstrating the edit operations and (2) the configuration and generation of the general transformation. This process is illustrated in Fig. 3, which is explained in the following step-by-step.

Phase 1: Modeling. In the first step, the user creates a model in the concrete syntax of the modeling language in her familiar modeling environment. This model comprises all model elements which are required to apply the transformation. The output of this step is called the *initial model*. In the second step, the user performs the complete transformation on this initial model by applying all necessary atomic operations, again in the concrete syntax. The output of this step is the *revised model* and a *change model* containing all changes that have been applied to the initial model during the demonstration. This change model together with the initial model and the revised model is the input for the second phase of the transformation specification process.

Phase 2: Configuration & Generation. In the second phase, an initial version of the transformation's *pre-* and *postconditions* is inferred by analyzing the initial model and the revised model, respectively. These automatically generated conditions from the example might not entirely express the intended pre- and postconditions of the transformation. Therefore, they only act as a basis for accelerating the transformation specification process and can be refined by the user in the next step and additional conditions may be added. After the revision of the conditions is finished, the *transformation* is generated from the change model and the revised pre- and postconditions.

3.2 Example

For exemplifying the presented MTBD process, a transformation for a simplified UML Class Diagram refactoring, namely "Extract Class" [15], is used. The aim of the refactoring is to create a new class and move the relevant attributes from an existing class into the new class.

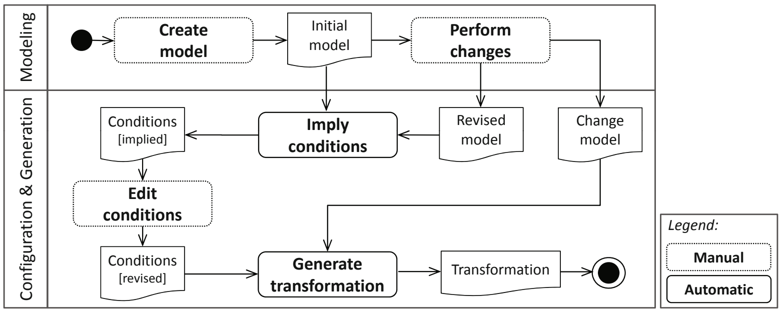


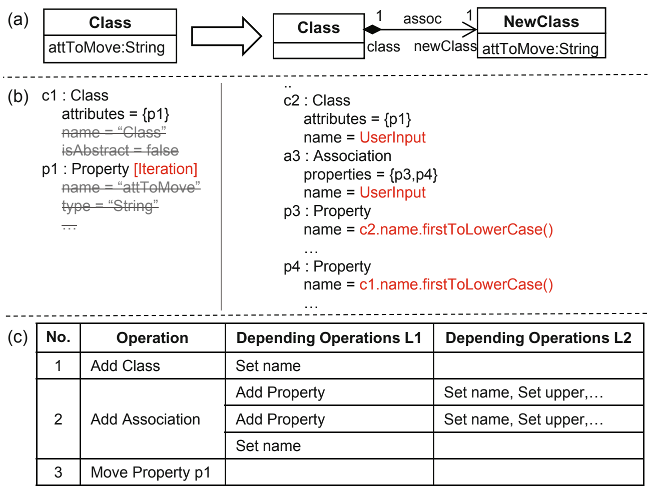
Fig. 3. MTBE Process for Endogenous Transformations

The transformation is demonstrated in Fig. 4(a). The user models the initial situation by introducing one class containing one attribute. Then, the user demonstrates the transformation by introducing another class and an association with two roles and, finally, the user moves the attribute to the newly introduced class. From this demonstration, the change model shown in Fig. 4(c) is obtained. For readability purposes, the changes are structured according to the container hierarchy of the model elements.

In addition to the change model, pre- and postconditions are derived from the initial and revised models, respectively. Subsequently, the user may fine-tune the inferred conditions by activating, deactivating, or modifying conditions as is depicted in Fig. 4(b). On the precondition side of our example, the name of a class and whether it is abstract or not does not matter. Furthermore, the transformation should be agnostic of the name and type of the attribute. Thus, these derived preconditions are deactivated. On the postcondition side, the user may introduce also some annotations for specifying how certain values in the resulting model should be obtained. In this context, a value may either be computed from values in the initial model, or by querying the user for an input value. In our example, the name of the newly introduced class is not derivable from the initial model and therefore has to be specified as *user input* before executing the transformation. The same is true for the association name. However, the role names should be derived from the existing model context. This is specified by additional expressions. In particular, the property name should be equal to the name of the adjacent class but the first letter has to be converted to lower case to fulfill common modeling conventions.

Besides fine-tuning the conditions, current MTBD approaches allow for annotating repetitions of certain edit operations. By this, the transformation may be configured to equally transform multiple model elements that fulfill the transformation's preconditions. In this example, such a mechanism is quite useful, because the transformation may then be capable of moving an arbitrary number of attributes from the original class to the newly introduced class.

An excerpt of the generated transformation is depicted in Listing 1.1. For simplicity, we just assume that the input parameters of the `extractClass` operation are specified by the user, e.g., by selecting the elements in the modeling



(b) c1 : Class
attributes = {p1}
~~name = “Class”~~
~~isAbstract = false~~
p1 : Property [Iteration]
~~name = “attToMove”~~
~~type = “String”~~
~~...~~

..
c2 : Class
attributes = {p1}
name = UserInput
a3 : Association
properties = {p3,p4}
name = UserInput
p3 : Property
name = c2.name.firstToLowerCase()
...
p4 : Property
name = c1.name.firstToLowerCase()
...

(c)

No.	Operation	Depending Operations L1	Depending Operations L2
1	Add Class	Set name	
2	Add Association	Add Property	Set name, Set upper,...
		Add Property	Set name, Set upper,...
		Set name	
3	Move Property p1		

Fig. 4. Example for Endogenous Transformations: (a) Demonstration, (b) Revised Conditions, and (c) Change Model

editor. Another scenario would be that the preconditions are employed to match all occurrences of the initial situation for a given model.

Listing 1.1. Generated Refactoring Code

```
method extractClass(String className, String attName,
       Class c, Collection<Property> props){

    //Check precondition
    assert c.attributes -> includesAll(props);

    //Create additional class
    Class newClass = new Class();
    newClass.setName(className);

    //Shift Attributes into new Class
    Iterator iter = props.iterator();
    while (iter.hasNext()){
        Property p = iter.hasNext();
        c.attributes().remove(p);
        newClass.attributes().add(p);
    }
    ... //Create additional elements and link them properly
}
```

The first statement is to verify that the preconditions are fulfilled by the given input elements. In this example, only one precondition has to be checked, namely if the selected attributes are all included in the feature `attributes` of the selected class. After checking the precondition, a new class is created and each attribute contained in the collection `props` is moved to the new class. Assuming that the user configured the transformation to support extracting multiple attributes at once, all changes applied to the attribute in the demonstration are repeated in a loop. Afterwards, the additional elements, particularly the association and the roles, have to be created and properly linked. Due to space limitations, this is not shown in the listing.

3.3 Existing Approaches

To the best of our knowledge, two MTBE approaches dedicated to endogenous transformations exist in literature. In the following we compare both approaches by highlighting their differences.

Brosch et al. [8,9] were the first to propose a "by-demonstration" approach to specify endogenous transformations. With this approach, endogenous transformations for any Ecore-based modeling language can be specified. Moreover, to be also independent from the used modeling editor, a *state-based model comparison* is employed to derive the atomic changes that have been performed on the initial model during the demonstration. The inherent imprecision of state-based model comparison is overcome by annotating unique identifiers to each model element before the user starts to demonstrate the transformation. By this, also element moves and intensively modified elements are supported. For expressing the pre- and postconditions, *OCL* constraints are employed. In the postconditions, users may also specify how attribute values in the target model shall be derived from values in the initial model. Repetitions of certain changes are realized by the notion of so-called *iterations*. Iterations are attached to precondition elements (representing model elements in the initial model) and indicate that each model element that fulfills these preconditions shall be transformed equally to the respective initial model element in the demonstration.

In the approach by Sun et al. [43], the changes applied during the demonstration are *recorded* and not derived by a subsequent comparison. After the demonstration, an inference engine generates a general transformation pattern which comprises the transformation's preconditions and its sequence of operations. This pattern may also be refined by the user in terms of adding preconditions and attribute value computations. In contrast to Brosch et al., *Groovy*[3]—a script language for the JVM—is employed to express these conditions and computations. In a more recent publication [42], Sun et al. extended this step so that users may also identify and annotate *generic operations*, which corresponds to the concept of iterations in [8]. However, these annotations are directly attached to the change model instead of to the preconditions as in [8].

4 MTBE for Exogenous Transformations

Various MTBE approaches [4,14,17,23,27,45,47], dedicated to exogenous transformations, have been proposed. Except [27] which is a *demonstration-based* approach, all others are based on *correspondences*. Thus, in the following, we discuss the general process of specifying exogenous transformations by-example based on correspondences, and subsequently, we present an instantiation of this

[3] `http://groovy.codehaus.org`

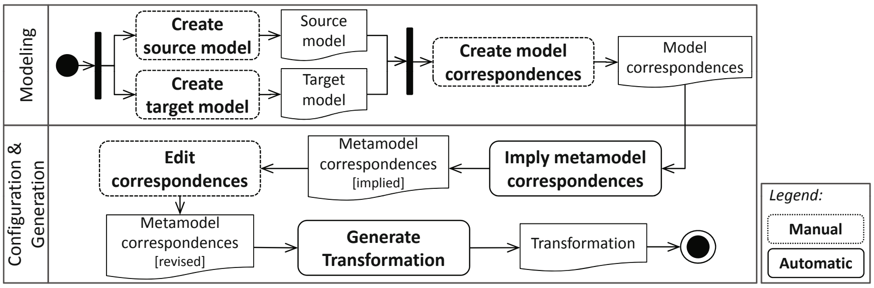


Fig. 5. MTBE Process for Exogenous Transformations

process for transforming UML Class Diagrams to ER Diagrams [11]. Finally, we conclude this section by elaborating on the peculiarities of current MTBE approaches.

4.1 Process

The main idea of MTBE for exogenous transformations is the semi-automatic generation of transformations from so-called correspondences between source and target model pairs. The underlying process for deriving exogenous model transformations from model pairs is depicted in Fig. 5. This process, which is largely the same for all existing approaches, consists of five steps grouped in two phases.

Phase 1: Modeling. In the first step, the user specifies semantically equivalent model pairs. Each pair consists of a source model and a corresponding target model. The user may decide whether she specifies a single model pair covering all important concepts of the modeling languages, or several model pairs whereby each pair focuses on one particular aspect. In the second step, the user has to align the source model and the target model by defining correspondences between source model elements and corresponding target model elements. For defining these correspondences, a correspondence language has to be available. One important requirement is that the correspondences may be established using the concrete syntax of the modeling languages. Hence, the modeling environment must be capable of visualizing the source and target models as well as the correspondences in one diagram or at least in one dedicated view.

Phase 2: Configuration & Generation. After finishing the mapping task, a dedicated reasoning algorithm is employed to automatically derive *metamodel correspondences* from the model correspondences. How the reasoning is actually performed is explained in more detail based on an example in Subsection 4.2. The automatically derived metamodel correspondences might not always reflect the intended mappings. Thus, the user may revise some metamodel correspondences or add further constraints and computations. Note that this step is not foreseen in all MTBE approaches, because it may be argued that this is contradicting with the general by-example idea of abstracting from the metamodels.

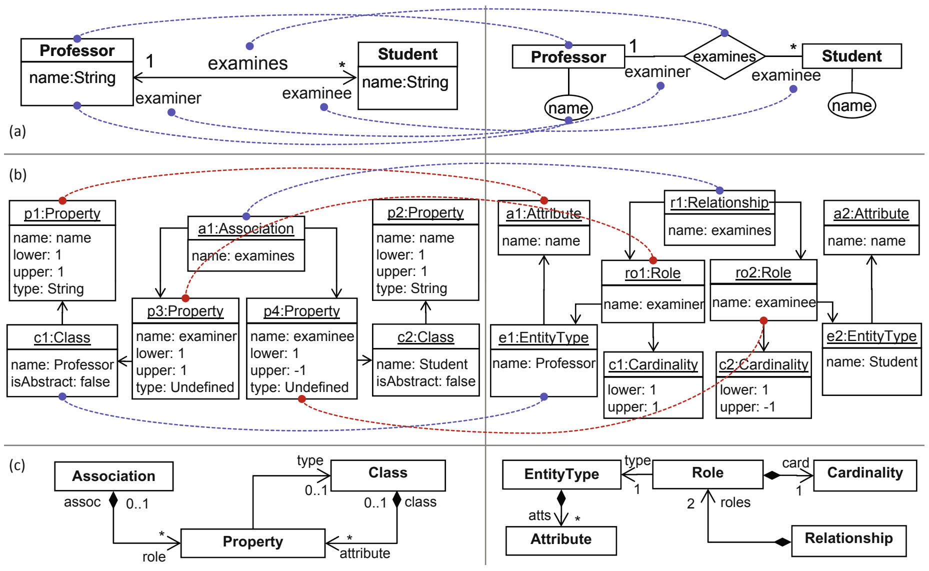


Fig. 6. Example for Exogenous Transformations: (a) Correspondences in concrete syntax, (b) Correspondences in abstract syntax, and (c) Metamodels

Nevertheless, it seems to be more user-friendly to allow the modification of the metamodel correspondences in contrast to modifying the generated model transformation code at the end of the generation process. Finally, a code generator takes the metamodel correspondences as input and generates executable model transformation code.

4.2 Example

For exemplifying the presented MTBE process, we now apply it to specify the transformation of the core concepts of UML class diagrams into ER diagrams. As modeling domain, a simple university information system is used. The user starts with creating the source model comprising the UML classes *Professor* and *Student* as well as a one-to-many association between them as depicted in the upper left area of Fig. 6. Subsequently, the corresponding ER diagram, depicted in the upper right area of Fig. 6, is created. In this figure, both models are represented in the concrete syntax as well as in the abstract syntax in terms of UML object diagrams. After both models are established, the correspondence model is created which consists of simple one-to-one mappings. These mappings are depicted as dashed lines in Fig. 6(a) and (b) between the source and target model elements.

In the next step, a reasoning algorithm now analyzes the model elements in the source and target models, i.e., objects, attribute values, and links, as well as the correspondences between them in order to derive metamodel correspondences. In the following, we discuss inferring metamodel correspondences between classes, attributes, and references.

Class correspondences. For detecting class correspondences, the reasoning algorithm first checks whether a certain object type in the source model is always mapped to the same object type in the target model. In this case, a *full equivalence mapping* between the respective classes is generated. In our example, a full equivalence mapping between objects of type `Class` and objects of type `EntityType` is inferred. However, `Properties` in the source model are mapped to different object types, namely `Attributes` and `Roles`, depending on their attribute values and links. For such cases, an additional mapping kind is used, namely *conditional equivalence mapping*. The conditions of such a mapping are derived by analyzing the links and values of the involved objects to find a discriminator for splitting the source objects into distinct sets having an unambiguous mapping to target objects. One appropriate heuristic for finding such a discriminator is to examine the container links of these objects. By this, the algorithm may deduce the constraints `property.class != null` to find an unambiguous mapping to `Attributes` and the condition `property.assoc != null` for `Roles`. Finally, also unmapped objects such as the `Cardinality` objects have to be considered. In our example, these objects have to be generated along with their container objects of type `Role`. Thus, the mapping for `Roles` has to be extended to a *one-to-two conditional equivalence mapping*. By this, a `Role` object and a properly linked `Cardinality` object is created for each `Property` in the source model.

Attribute correspondences. Generally, attributes in metamodels may be distinguished in *ontological attributes* and *linguistic attributes* [20]. Ontological attributes represent semantics of the real-world domain. Values have to be explicitly given by the user. Examples for ontological attributes are `Class.name` or `Attribute.name`. In order to find correspondences between ontological attributes, heuristics have to be used which compare the attribute values, for instance, based on edit distance metrics. In our example, we may conclude that `Class.name` should be mapped to `EntityType.name` because the values of the `name` attributes are equivalent for each `Class`/`EntityType` object pair. In contrast, linguistic attributes are used for the reification of modeling concepts such as `Class.isAbstract`. Usually these attributes have predefined, restricted value ranges in the language definition. When dealing with linguistic attributes in the context of MTBE, similar heuristics based on string matching as for ontological attributes may be used. However, the probability for accidentally matching wrong pairs and for ambiguities is much higher. Consider for instance the mapping between the property `p3` and the role `ro1` without taking into account other mappings. Then, we cannot decide if the attribute `Property.lower` is mapped to `Role.cardinality.lower` or to `Role.cardinality.upper` by solely looking at the example. Here, the problem is that we do not have unique values which help us finding the metamodel correspondences. This may be improved by using matching techniques on the metamodel level for finding similarities between attribute names. An alternative solution used in this example is to define an additional mapping between the property `p4` and the role `ro2` where we have unique values for the lower and upper attributes.

Reference correspondences. For deriving reference correspondences, the afore calculated class correspondences are of paramount importance since they serve as anchors for reasoning about corresponding links. For example, consider the reference `atts` in the ER metamodel between `EntityType` and `Attribute`. For finding the corresponding reference in the UML metamodel, we have to reason about the previously derived class correspondences. First, the `Attribute` class in the ER metamodel is mapped to the `Property` class of the UML metamodel. Furthermore, when looking at the example models, each `Attribute` is contained by an `EntityType` and each `Property` is contained by a `Class`. Luckily, the `EntityType` class is accordingly mapped to the `Class` class on the metamodel level, so that we can conclude that whenever transforming a `Property` into an ER `Attribute`, a link between the created `Attribute` and the `EntityType` previously generated for the `Class` containing the aforementioned `Property` is generated. Thus, there should be a correspondence between the reference `atts` in the ER metamodel and the reference `attribute` in the UML metamodel.

After the metamodel correspondences have been derived automatically, MTBE approaches usually allow the user to verify and adapt the generated correspondences. For our running example however, this is not required. The next task is to automatically translate the correspondences into executable transformation code. Listing 1.2 depicts the transformation required for our running example in imperative OCL [10]. For each metamodel correspondence, a transformation rule is generated which queries the source model and generates the corresponding target model elements. Inside each rule, the attribute and reference correspondences are translated to assignments. Please note that current transformation engines are able to schedule rules automatically and to build an implicit trace model between the source and target model. Based on this trace model, assignments such as `e.atts = c.attribute` (cf. line 2 in Listing 1.2) are automatically resolved. In particular, not the UML attributes (*c.attribute*) are assigned to the `EntityType`, but the ER attributes generated from these UML attributes are resolved by applying the trace model. These features of transformation languages and their encompassing engines drastically ease the transformation code generation from correspondences.

Listing 1.2. Generated Transformation Code

```
 1 rule 1: Class.allInstances() -> foreach (c |
 2     create Entity e (e.name = c.name, e.atts = c.attribute);
 3 rule 2: Property.allInstances()
 4     -> select(p | p.class <> OclUndefined) -> foreach( p |
 5     create Attribute a (a.name = p.name));
 6 rule 3: Property.allInstances() -> select(p |
 7     p.assoc <> OclUndefined) -> foreach( p |
 8     create Role r (r.name = p.name, r.cardinality = c),
 9     create Cardinality c (c.upper = p.upper, c.lower = p.lower,
10     r.type = p.type));
11 rule 4: Association.allInstances() -> foreach(a |
12     create Relationship r (r.name = a.name, r.roles = a.role));
```

4.3 Existing Approaches

We now compare existing approaches by highlighting their commonalities and differences. Mostly all approaches define the input for deriving exogenous transformations as a triple comprising an input model, a semantically equivalent output model as well as correspondences between these two models. These models have to be built by the user, preferably using the concrete syntax as is, e.g., supported by [47], but most approaches do not provide dedicated support for defining the correspondences in graphical modeling editors.

Langer et al. [27] presented, in contrast to the correspondence-based approaches, a demonstration-based approach which allows to demonstrate transformation rules incrementally by giving for each rule an input model fragment and a corresponding output model fragment so that the correspondences between the fragments can be automatically inferred and do not have to be manually specified in advance.

Subsequently, reasoning techniques such as specific rules again implemented as model transformations [17,27,45,47], inductive logic [4], and relational concept analysis [14] are used to derive model transformation code. Current approaches support the generation of graph transformation rules [4,45] or ATL code [17,27,47].

All approaches aim for *semi-automated* transformation generation meaning that the generated transformations are intended to be further refined by the user. This is especially required for transformations involving global model queries and attribute calculations such as aggregation functions, which have to be manually added. Furthermore, it is recommended to iteratively develop the transformations, i.e., after generating the transformations from initial examples, the examples must be adjusted or the transformation rules must be adapted in case the actual generated output model is not fully equivalent to the expected output model. However, in many cases it is not obvious whether to adapt *the aligned examples* or the *generated transformations*. Furthermore, adjusting the examples might be a tedious process requiring a large number of transformation examples to assure the quality of the inferred rules. In this context, self-tuning transformations have been introduced [22,23]. Self-tuning transformations employ the examples as training instances in an iterative process for further improving the quality of the transformation. The goal is to minimize the differences between the actual output model produced by the transformation and the expected output model given by the user by using the differences to adapt the transformation over several iterations. Of course, adapting the transformation is a computation intensive problem leading to very large search spaces. While in [22] domain-specific search space pruning tailored to EMF-based models is used, a generic meta-heuristic based approach is used in [23] to avoid an exhaustive search.

5 Lessons Learned and Future Challenges

In this section, some lessons learned from applying and developing MTBE approaches during the last 5 years are summarized.

		MTBE Technique	
		Correspondences	***Demonstrations***
Transformation Scenario	***Endogenous Transformations***		[8,42]
	Exogenous Transformations	[4,14,17,23,45,47]	[27]

Fig. 7. Classification of MTBE approaches by whether they support exogenous or endogenous transformations and whether they are correspondence or demonstration-based

Different Transformation Scenarios/Different MTBE Techniques. When categorizing MTBE approaches w.r.t. transformation scenarios and MTBE techniques (cf. Fig 7), the following discriminators are derivable. Approaches using correspondences are exclusively but intensively applied for deriving exogenous transformations. Surprisingly, not a single work considers to apply correspondences for endogenous transformations. In contrast, demonstration-based approaches originally have been proposed for endogenous transformations and only one work discusses the application of demonstrations to derive exogenous transformations.

Challenge: *What are the commonalities and differences of correspondence-based and demonstration-based approaches?*

MTBE as Enabler for Test-driven Transformation Development. A significant advantage of MTBE is the existence of examples. Besides serving as input for the derivation of a model transformation during the MTBE process, the example models may also be used to *test the generated transformation.* By applying the inferred transformation again to the source model, the obtained target model may be compared to the target model specified during the MTBE process. If any differences are found in the comparison, either the transformation or the target model is obviously wrong. In this sense, MTBE inherently implements the idea of *test-driven development* [5]. An interesting direction for future work in this area is to automatically suggest corrections to the transformation based on the detected differences between the target example model and the actual transformation result.

Challenge: *Which logic and machine learning techniques can be employed for optimizing the quality of derived transformations in reasonable time with a small amount of examples?*

MTBE outperforms Metamodel Matching. With the rise of the Semantic Web and the emerging abundance of ontologies, several automated matching approaches and tools have been proposed (cf. [37,40] for an overview). The typical output of such tools are correspondences mostly computed based on schema information, e.g., name and structure similarity. In experiments, we have reused ontology matching tools for matching metamodels by beforehand transforming metamodels into corresponding ontologies. However, the quality of the produced correspondences is on average significantly lower compared to MTBE approaches [21]. The reasons for this are twofold. First, structural heterogeneities between metamodels and the mismatch between the terminology used for different modeling languages makes it hard to reason about correspondences solely on the metamodel level. Second, there is no automated evaluation of the quality of correspondences based on the model level, because the matching approaches are not bound to a specific integration scenario [6], such as transformation, merge, or search. Finally, we also learned that the preparation phase required for using MTBE approaches, i.e., building the example models, is less work than the comprehensive reworking phase, i.e, validating and correcting the correspondences, required for metamodel matching approaches.

Challenge: *More empirical studies on MTBE approaches for identifying the strengths and weaknesses of existing approaches are required.*

Multifaceted Usage of Examples. Another benefit of specifying endogenous model transformations by demonstration is to reuse the developed transformation specifications for *detecting applications of the transformation*. Since such a specification developed using an MTBD approach comprises the transformation's preconditions, postconditions, and its change pattern, a dedicated detection mechanism may triage arbitrary model differences for the transformation's change pattern and, given the pattern could be found, validate its pre- and postconditions to reveal an application of the transformation. This is especially useful if these transformations implement model refactorings because this knowledge gains valuable information on the evolution of a model and is of paramount importance for various application domains such as model co-evolution, model versioning, and model repository mining.

Challenge: *How may the developed examples and derived transformations be employed for supporting different model management tasks in MDE?*

6 Resume

More than 30 papers have been published in the first 5 years and more and more research groups start working in this area. MDE and by-example approaches both aim to ease the development of software systems. However, both stand on orthogonal dimensions. MDE aims to abstract from the implementation level of software systems such as particular technology platforms and programming languages by using platform independent modeling techniques. In contrast, by-example approaches aim to ease the development of systems by using examples instead of directly developing generalized programs. We believe that combining both paradigms seems to be promising and would have a major impact on end-user programming or better say end-user modeling.

References

1. Amelunxen, C., Königs, A., Rötschke, T., Schürr, A.: MOFLON: A Standard-Compliant Metamodeling Framework with Graph Transformations. In: Rensink, A., Warmer, J. (eds.) ECMDA-FA 2006. LNCS, vol. 4066, pp. 361–375. Springer, Heidelberg (2006)
2. Atkinson, C., Kühne, T.: Model-Driven Development: A Metamodeling Foundation. IEEE Software 20(5), 36–41 (2003)
3. Baar, T., Whittle, J.: On the Usage of Concrete Syntax in Model Transformation Rules. In: Virbitskaite, I., Voronkov, A. (eds.) PSI 2006. LNCS, vol. 4378, pp. 84–97. Springer, Heidelberg (2007)
4. Balogh, Z., Varró, D.: Model transformation by example using inductive logic programming. Software and System Modeling 8(3), 347–364 (2009)
5. Beck, K.: Test Driven Development: By Example. Addison-Wesley (2002)
6. Bernstein, P.A., Melnik, S.: Model management 2.0: manipulating richer mappings. In: Int. Conf. on Management of Data (SIGMOD 2007), pp. 1–12. ACM (2007)
7. Bézivin, J.: On the unification power of models. Software and System Modeling 4(2), 171–188 (2005)
8. Brosch, P., Langer, P., Seidl, M., Wieland, K., Wimmer, M., Kappel, G., Retschitzegger, W., Schwinger, W.: An Example Is Worth a Thousand Words: Composite Operation Modeling By-Example. In: Schürr, A., Selic, B. (eds.) MODELS 2009. LNCS, vol. 5795, pp. 271–285. Springer, Heidelberg (2009)
9. Brosch, P., Langer, P., Seidl, M., Wimmer, M.: Towards End-User Adaptable Model Versioning: The By-Example Operation Recorder. In: Proc. of CVSM 2009 @ ICSE 2009. IEEE (2009)
10. Cabot, J.: From Declarative to Imperative UML/OCL Operation Specifications. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 198–213. Springer, Heidelberg (2007)
11. Chen, P.P.S.: The Entity-Relationship Model—Toward a Unified View of Data. ACM Transactions on Database Systems 1, 9–36 (1976)
12. Cuadrado, J.S., Molina, J.G., Tortosa, M.M.: RubyTL: A Practical, Extensible Transformation Language. In: Rensink, A., Warmer, J. (eds.) ECMDA-FA 2006. LNCS, vol. 4066, pp. 158–172. Springer, Heidelberg (2006)
13. Czarnecki, K., Helsen, S.: Feature-based survey of model transformation approaches. IBM Systems Journal 45(3), 621–646 (2006)
14. Dolques, X., Huchard, M., Nebut, C.: From transformation traces to transformation rules: Assisting Model Driven Engineering approach with Formal Concept Analysis. In: 17th Int. Conf. on Conceptual Structures (ICCS 2009), vol. 483, pp. 15–29. CEUR-WS (2009)
15. Fowler, M.: Refactoring: Improving the Design of Existing Code. Addison-Wesley, Boston (1999)
16. France, R., Rumpe, B.: Model-driven Development of Complex Software: A Research Roadmap. In: 29th Int. Conf. on Software Engineering (ICSE 2007) - Future of Software Engineering, pp. 37–54 (2007)
17. García-Magariño, I., Gómez-Sanz, J.J., Fuentes-Fernández, R.: Model Transformation By-Example: An Algorithm for Generating Many-to-Many Transformation Rules in Several Model Transformation Languages. In: Paige, R.F. (ed.) ICMT 2009. LNCS, vol. 5563, pp. 52–66. Springer, Heidelberg (2009)
18. ISO/IEC: 14977:1996(E) Information technology – Syntactic metalanguage – Extended BNF, International standard (1996)

19. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: A model transformation tool. Sci. Comput. Program 72(1-2), 31–39 (2008)
20. Kappel, G., Kapsammer, E., Kargl, H., Kramler, G., Reiter, T., Retschitzegger, W., Schwinger, W., Wimmer, M.: Lifting Metamodels to Ontologies: A Step to the Semantic Integration of Modeling Languages. In: Wang, J., Whittle, J., Harel, D., Reggio, G. (eds.) MoDELS 2006. LNCS, vol. 4199, pp. 528–542. Springer, Heidelberg (2006)
21. Kappel, G., Kargl, H., Kramler, G., Schauerhuber, A., Seidl, M., Strommer, M., Wimmer, M.: Matching Metamodels with Semantic Systems - An Experience Report. In: Workshop Proceedings of Datenbanksysteme in Business, Technologie und Web, BTW 2007 (2007)
22. Kargl, H., Wimmer, M., Seidl, M., Kappel, G.: SmartMatcher: Improving Automatically Generated Transformations. Datenbank-Spektrum 29, 42–52 (2009)
23. Kessentini, M., Sahraoui, H.A., Boukadoum, M.: Model Transformation as an Optimization Problem. In: Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., Völter, M. (eds.) MODELS 2008. LNCS, vol. 5301, pp. 159–173. Springer, Heidelberg (2008)
24. Kleppe, A.: Software Language Engineering: Creating Domain-Specific Languages Using Metamodels. Addison-Wesley (2008)
25. Kolovos, D.S., Paige, R.F., Polack, F.A.C.: The Epsilon Transformation Language. In: Vallecillo, A., Gray, J., Pierantonio, A. (eds.) ICMT 2008. LNCS, vol. 5063, pp. 46–60. Springer, Heidelberg (2008)
26. Kühne, T.: Matters of (Meta-)Modeling. Software and System Modeling 5(4), 369–385 (2006)
27. Langer, P., Wimmer, M., Kappel, G.: Model-to-Model Transformations By Demonstration. In: Tratt, L., Gogolla, M. (eds.) ICMT 2010. LNCS, vol. 6142, pp. 153–167. Springer, Heidelberg (2010)
28. de Lara, J., Vangheluwe, H.: AToM^3: A Tool for Multi-formalism and Meta-modelling. In: Kutsche, R.-D., Weber, H. (eds.) FASE 2002. LNCS, vol. 2306, pp. 174–188. Springer, Heidelberg (2002)
29. Lieberman, H.: Your wish is my command: programming by example. Morgan Kaufmann Publishers Inc. (2001)
30. Ma, H., Shao, W.-Z., Zhang, L., Ma, Z.-Y., Jiang, Y.-B.: Applying OO Metrics to Assess UML Meta-models. In: Baar, T., Strohmeier, A., Moreira, A., Mellor, S.J. (eds.) UML 2004. LNCS, vol. 3273, pp. 12–26. Springer, Heidelberg (2004)
31. Mens, T., Gorp, P.V.: A Taxonomy of Model Transformation. Electr. Notes Theor. Comput. Sci. 152, 125–142 (2006)
32. Nickel, U., Niere, J., Zündorf, A.: The FUJABA environment. In: Int. Conf. on Software Engineering (ICSE 2000), pp. 742–745 (2000)
33. Object Management Group (OMG): Meta Object Facility, Version 2.0 (2006), `http://www.omg.org/spec/MOF/2.0/PDF/`
34. Object Management Group (OMG): Unified Modeling Language Superstructure Specification, Version 2.1.2 (2007), `http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF`
35. OMG, O.: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. Final Adopted Specification (November 2005)
36. OMG, O.: OCL Specification Version 2.0 (June 2005), `http://www.omg.org/docs/ptc/05-06-06.pdf`
37. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB Journal 10(4), 334–350 (2001)
38. Schmidt, D.C.: Model-Driven Engineering. IEEE Computer 39(2), 25–31 (2006)

39. Sendall, S., Kozaczynski, W.: Model Transformation: The Heart and Soul of Model-Driven Software Development. IEEE Software 20, 42–45 (2003)
40. Shvaiko, P., Euzenat, J.: A Survey of Schema-Based Matching Approaches. In: Spaccapietra, S. (ed.) Journal on Data Semantics IV. LNCS, vol. 3730, pp. 146–171. Springer, Heidelberg (2005)
41. Strommer, M., Wimmer, M.: A Framework for Model Transformation By-Example: Concepts and Tool Support. In: 46th Int. Conf. on Objects, Components, Models and Patterns (TOOLS 2008). LNBIP, vol. 11, pp. 372–391. Springer, Heidelberg (2008)
42. Sun, Y., Gray, J., White, J.: MT-Scribe: an end-user approach to automate software model evolution. In: 33rd Int. Conf. on Software Engineering (ICSE 2011), pp. 980–982. ACM (2011)
43. Sun, Y., White, J., Gray, J.: Model Transformation by Demonstration. In: Schürr, A., Selic, B. (eds.) MODELS 2009. LNCS, vol. 5795, pp. 712–726. Springer, Heidelberg (2009)
44. Taentzer, G.: AGG: A Graph Transformation Environment for Modeling and Validation of Software. In: Pfaltz, J.L., Nagl, M., Böhlen, B. (eds.) AGTIVE 2003. LNCS, vol. 3062, pp. 446–453. Springer, Heidelberg (2004)
45. Varró, D.: Model Transformation by Example. In: Wang, J., Whittle, J., Harel, D., Reggio, G. (eds.) MoDELS 2006. LNCS, vol. 4199, pp. 410–424. Springer, Heidelberg (2006)
46. Whittle, J., Moreira, A., Araújo, J., Jayaraman, P.K., Elkhodary, A.M., Rabbi, R.: An Expressive Aspect Composition Language for UML State Diagrams. In: Engels, G., Opdyke, B., Schmidt, D.C., Weil, F. (eds.) MODELS 2007. LNCS, vol. 4735, pp. 514–528. Springer, Heidelberg (2007)
47. Wimmer, M., Strommer, M., Kargl, H., Kramler, G.: Towards Model Transformation Generation By-Example. In: 40th Hawaiian Int. Conf. on Systems Science (HICSS 2007). IEEE Computer Society (2007)
48. Zloof, M.M.: Query-by-Example: the Invocation and Definition of Tables and Forms. In: Int. Conf. on Very Large Data Bases (VLDB 1975), pp. 1–24. ACM (1975)

On Computing the Importance of Associations in Large Conceptual Schemas

Antonio Villegas, Antoni Olivé, and Maria-Ribera Sancho

Department of Service and Information System Engineering
Universitat Politècnica de Catalunya
Barcelona, Spain
{avillegas,olive,ribera}@essi.upc.edu

Abstract. The visualization and the understanding of large conceptual schemas require the use of specific methods. These methods generate clustered, summarized or focused schemas that are easier to visualize and to understand. All of these methods require computing the importance of the elements in the schema but, up to now, only the importance of entity types has been taken into account. In this paper, we present three methods for computing the importance of associations by taking into account the knowledge defined in the structural and behavioral parts of the schema. We experimentally evaluate these methods with large real-world schemas and present the main conclusions we have drawn from the experiments.

Keywords: Conceptual Modeling, Importance, Associations.

1 Introduction

A conceptual schema defines the general knowledge about the domain that an information system of an organization needs to know to perform its functions [16,20,18]. The conceptual schema of many real-world information systems are too large to be easily managed or understood. One of the most challenging and long-standing goals in conceptual modeling is to ease the comprehension of large conceptual schemas [11,15]. The visualization and understanding of these schemas requires the use of specific methods, which are not needed in small schemas. These methods generate indexed [24,5,23,22], clustered [10,12,19], summarized [8,28,27] or focused [26,25] schemas that are easier to visualize and to understand.

Many of the above methods require computing the importance (also called relevance or score) of each element in the schema. The computed importance induces an ordering of the elements, which plays a key role in the steps and result (output) of the method. Up to now, the existing metrics of importance for schema elements were mainly centered in computing the importance of entity types, but not in the importance of associations.

The main objective of this paper is to analyze existing metrics for measuring the importance of entity types, and then to adapt them to be able to work with associations. We present two different methods to compute the importance of associations inspired by the entity-type importance methods of occurrence counting and link analysis [24]. In addition, our contribution also includes a method to obtain the importance of associations by adapting a betweenness-centrality measure [2,3] from the fields of graph theory and complex networks [1].

Our approach takes into account the knowledge defined in the schema about associations, including their participant entity types, the cardinality constraints of those participations, the general constraints of the schema, and the specification of behavioral events. All of them contribute to measure the importance of associations.

We have experimentally evaluated each method using the conceptual schemas of the osCommerce [21] and EU-Rent [9], the UML2 metaschema [13], a fragment of the HL7 schemas [26], and the OpenCyc ontology [6]. All of them contain a large amount of entity types and associations, which make difficult their understanding. We analyze the differences between methods and schemas, and make conclusions on the importance of associations and its value in the comprehension of large conceptual schemas.

The rest of the paper is organized as follows. Section 2 introduces the basic concepts and notations. Section 3 reviews the concept of reification of associations. Section 4 presents the methods to compute the importance of associations in detail. Section 5 describes the experimentation with the methods, the results obtained, and the conclusions we have drawn. Finally, Section 6 summarizes the paper and points out future work.

2 Basic Concepts and Notations

In this section we review the main concepts and the notations we have used to define the knowledge of conceptual schemas. In this paper, we deal with schemas written in the UML/OCL[13,14], which consists of the elements summarized in Def. 1.

Definition 1. *(Conceptual Schema) A conceptual schema $\mathcal{CS}$ is defined as a triple $\mathcal{CS} = \langle \mathcal{E}, \mathcal{R}, \mathcal{C} \rangle$, where:*

- *$\mathcal{E}$ is a set of* entity types. *Some $e \in \mathcal{E}$ represents event types [17].*
- *$\mathcal{R}$ is a set of* associations *between entity types of $\mathcal{E}$. The degree of an association is the number of entity types that participate on it. An association r has a degree $n \geq 2$.*
- *$\mathcal{C}$ is a set of* schema rules. *$\mathcal{C}$ contains textual OCL constraints and the transformation of all graphical UML constraints, including cardinality constraints, into OCL expressions.*

Table 1 summarizes the basic metrics used in the rest of the paper. If $r \in \mathcal{R}$ then $members(r)$ denotes the set of entity types that participate in the association r, and $assoc(e)$ the set of associations in which e participates. Note that an entity type e may participate more than once in the same association, and therefore $members(r)$ and $assoc(e)$ are multisets (may contain duplicate elements). Moreover, $conn(e)$ denotes the multiset of entity types connected to e through associations.

Table 1. Definition of basic metrics

Notation	Definition
$members(r)$	$= \{e \in \mathcal{E} \mid e \text{ is a participant of } r\}$
$assoc(e)$	$= \{r \in \mathcal{R} \mid e \in members(r)\}$
$conn(e)$	$= \uplus_{r \in assoc(e)} \{members(r) \backslash \{e\}\}$[1]

We denote by $\mathcal{C}$ the set of schema rules of a conceptual schema, including constraints, derivation rules and pre- and postconditions. Each schema rule c is defined in the context of an entity type, denoted by $context(c)$. In OCL, each rule c consists of a set of OCL expressions (see OCL [14]) that may refer to several entity types which are denoted by $ref(c)$. We also include in $\mathcal{C}$ the schema rules corresponding to the equivalent OCL invariants of the cardinality constraints.

Table 2. Definition of extended metrics

Notation	Definition
$ref(c)$	$= \{e \in \mathcal{E} \mid e \text{ is referenced in } c \in \mathcal{C}\}$
$links_{context}(c)$	$= \{\{e, e'\} \mid e, e' \in \mathcal{E} \wedge e = context(c) \wedge e' \in ref(c)\}$
$links_{nav}(c)$	$= \{\{e, e'\} \mid e, e' \in \mathcal{E} \wedge e \rightarrow e' \text{ is a navigation in } c\}$
$links(c)$	$= links_{context}(c) \cup \ links_{nav}(c)$
$rconn(e)$	$= \uplus_{c \in \mathcal{C}} \{e' \in \mathcal{E} \mid \{e, e'\} \subset links(c)\}$

A special kind of OCL expression is the navigation expression that define a schema navigation from an entity type to another through an association (see `NavigationCallExp` of OCL in [14]). Such expressions only contain two entity types as its participants, i.e. the source entity type and the target one ($e \rightarrow e'$). We denote by $links_{nav}(c)$ the set of pairs $\langle e, e' \rangle$ that participate in the navigation expressions of c. We also denote by $links_{context}(c)$ the sets of pairs of entity types composed by the context of the rule c and every one of the participant entity types of such rule ($e \in ref(c)$). Finally, we define $links(c)$ as the union of $links_{context}(c)$ with $links_{nav}(c)$ and, $rconn(e)$ as the multiset of entity types that compose a pair with e in $links(c)$. Note that since we use $\uplus$, $rconn(e)$ may contain duplicates because it takes into account each rule c, and an entity type e can be related to another one e' in two or more different rules.

[1] Note that "$\backslash$" denotes the difference operation of multisets as in $\{a, a, b\} \backslash \{a\} = \{a, b\}$ and "$\uplus$" denotes the multiset (or bag) union that produces a multiset as in $\{a, b\} \uplus \{a\} = \{a, a, b\}$.

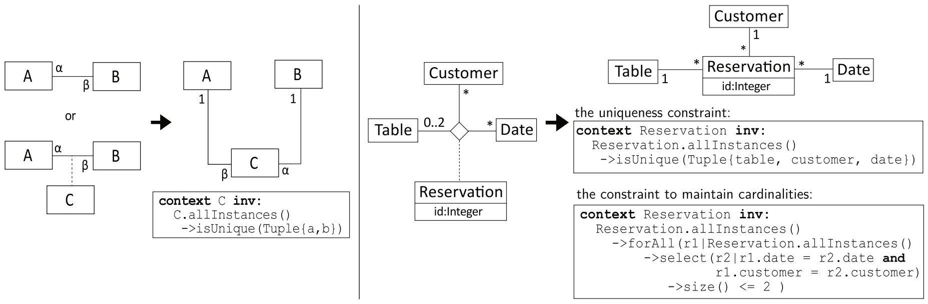
Intuitively, $rconn(e)$ is the multiset of entity types to which an entity type e is connected through schema rules. Table 2 shows the formal definition of the extended metrics for schema rules.

3 Reifications

Reifying an association consists in viewing it as an entity type [16]. When we view an association r as an entity e_r, we say that the entity reifies the association. The reification of an association does not add any new knowledge to a schema, but it is an interesting schema transformation that can be used in our context of importance of associations. Since the existing methods for importance computing are defined for entity types, the reification of association into entity types is a way to easily adapt the schema and such methods in order to compute the importance of associations.

Formally, the reification of an association $r \in \mathcal{R}$ is an entity type e_r connected with the participant entity types of r through *intrinsic* binary associations. For the case of association classes, since an association class is also an entity type, we make an implicit reification only changing the connections with the participants by adding the intrinsic binary associations.

In the case of a binary association r (left side of Fig. 1), the cardinality constraints after the transformation are placed in the sides of the entity type e_r that reifies the association. The cardinality constraints of a participant (e.g. α of A) go to the new intrinsic association between the other participant (B) and the new entity (C), placed on the side of the new entity type. On the side of the previous participants (A and B) the cardinality equals 1. Furthermore, to maintain all the semantics we also need a new *uniqueness* constraint expressed in OCL, as shown in Fig. 1.



```
context C inv:
  C.allInstances()
    ->isUnique(Tuple{a,b})
```

the uniqueness constraint:

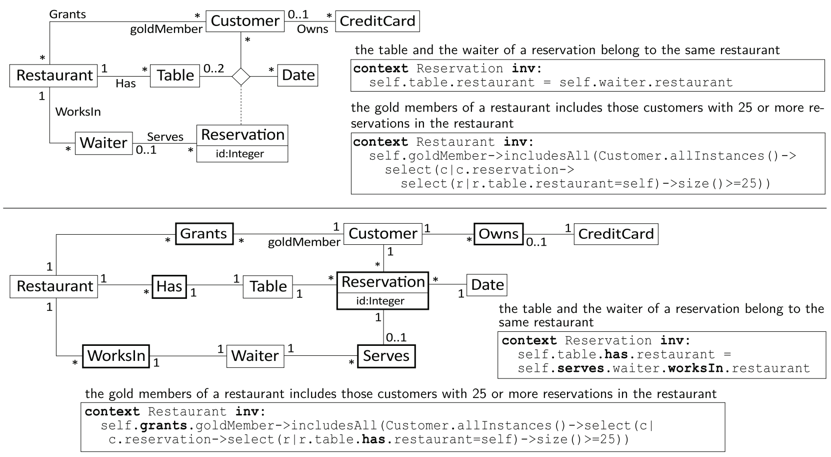
```
context Reservation inv:
  Reservation.allInstances()
    ->isUnique(Tuple{table, customer, date})
```

the constraint to maintain cardinalities:

```
context Reservation inv:
  Reservation.allInstances()
    ->forAll(r1|Reservation.allInstances()
        ->select(r2|r1.date = r2.date and
                  r1.customer = r2.customer)
           ->size() <= 2 )
```

Fig. 1. Reification of binary (left side) and ternary (right side) associations

For the case of an n-ary association the transformation is similar (see right side of Fig. 1). However, the multiplicities in the intrinsic binary associations between the participants and the entity type that reifies the n-ary association are always "1" on the participant side and "*" on the entity-type side. In this case a uniqueness constraint is also needed and follows the same idea as with binary associations. Furthermore, the cardinality constraints of the initial association must be expressed as an OCL constraint to maintain all the semantics in the reified version. For example, the multiplicity "0..2" in the *Table* participant of Fig. 1 (right side) is transformed into the OCL constraint shown at the bottom.



the table and the waiter of a reservation belong to the same restaurant

```
context Reservation inv:
  self.table.restaurant = self.waiter.restaurant
```

the gold members of a restaurant includes those customers with 25 or more reservations in the restaurant

```
context Restaurant inv:
  self.goldMember->includesAll(Customer.allInstances()->
    select(c|c.reservation->
      select(r|r.table.restaurant=self)->size()>=25))
```

the table and the waiter of a reservation belong to the same restaurant

```
context Reservation inv:
  self.table.has.restaurant =
  self.serves.waiter.worksIn.restaurant
```

the gold members of a restaurant includes those customers with 25 or more reservations in the restaurant

```
context Restaurant inv:
  self.grants.goldMember->includesAll(Customer.allInstances()->select(c|
   c.reservation->select(r|r.table.has.restaurant=self)->size()>=25))
```

Fig. 2. A fragment of conceptual schema (up) and its version with reifications (down)

After reifying all the associations in a schema $\mathcal{CS} = \langle \mathcal{E}, \mathcal{R}, \mathcal{C} \rangle$ we obtain a new schema $\mathcal{CS}^+ = \langle \mathcal{E}^+, \mathcal{R}^+, \mathcal{C}^+ \rangle$. The set of entity types $\mathcal{E}^+$ contains the entity types $\mathcal{E}$ of $\mathcal{CS}$, and the entity types $\mathcal{E}_\mathcal{R}$ from the reifications of the initial associations ($\mathcal{E}^+ = \mathcal{E} \cup \mathcal{E}_\mathcal{R}$). The associations in $\mathcal{R}^+$ are the binary intrinsic associations that connect each $e_r \in \mathcal{E}_\mathcal{R}$ with its participants $e \in \mathcal{E}$. Finally, $\mathcal{C}^+$ contains the same constraints as in $\mathcal{C}$ including the required navigation changes to match the new schema $\mathcal{CS}^+$, and the uniqueness constraints introduced in the reifications.

Figure 2 (top) shows an example of conceptual schema with 6 associations (one of them, an association class of degree 3) and two OCL constraints that describes a fragment of an information system for reservations in restaurants. Figure 2 (bottom) shows the same schema with reifications (marked with bold rectangles). Note that the two constraints include the new navigations (bold text) to match with the new schema. Basically, a navigation $e \rightarrow e'$ has to be changed by including the navigation to the entity type that reifies the association in the middle of the expression, producing a navigation like $e \rightarrow e_r \rightarrow e'$. The reification of the schema produces 6 additional uniqueness constraints and a constraint to preserve the cardinality 0..2 in *Table* as indicated in Fig. 1. These 7 constraints are not shown in Fig. 2 for the sake of simplicity.

4 Methods for Computing the Importance

In this section we present the definition of three methods for computing the importance of associations in a schema. The methods are an adaptation of methods that compute the importance of entity types to associations. Each method is followed by a brief description showing its details, and its application to the example of Fig. 2. We apply the methods to both entity types and associations, but the focus of the paper is on associations. Concretely, these methods require to be applied to the schema with reifications (bottom of Fig. 2) in order to compute the importance of the entity types e_r that are reifications of associations. Then, such importance is directly the importance of each association $r \in \mathcal{R}$ of the original schema because each e_r is the representation of an association r in the schema with reifications.

4.1 Occurrence Counting

There exist different kinds of methods to compute the importance of elements in a schema. The simplest family of methods is that based on *occurrence counting* [5,12,22], where the importance of a schema element is equal to the number of characteristics it has in the schema. Therefore, the more characteristics about an element, the more important it will be.

Our approach adapts the occurrence counting methods described in [24] to compute the importance of associations taking into account the characteristics an association may have in a schema. Those include the number of participants, its multiplicities, and its usage in OCL expressions to navigate the schema. Formally, the importance $\mathcal{OC}(r)$ of an association $r \in \mathcal{R}$ is defined as,

$$\mathcal{OC}(r) = |conn(e_r)| + |rconn(e_r)|$$

where the previous metrics for entity types are applied to the entity type e_r that appears from the reification of r, as explained in Sec. 3. Concretely, $|conn(e_r)|$ is the number of associations where e_r participates, which can be easily mapped to the number of connections that the original association r has in the schema. In the same way, $|rconn(e_r)|$ indicates the usage in OCL expressions of the entity type e_r, and consequently, the usage of r.

Table 3 shows the results of applying the occurrence method to the example in Fig. 2. Note that *Reservation* is the association that has a greater importance due to its number of participants and its usage in OCL navigations. As expected, the association *Owns* has a lower importance because it does not participate in any OCL expression apart from the uniqueness constraints of its reification (as shown in Fig. 1 for a binary association).

We also show in Tab. 3 the importance of entity types computed with the same method (the sum of $|conn(e)|$ and $|rconn(e)|$). *Restaurant* is the most important entity types according to the method, followed by *Customer* and *Table.*

4.2 Link Analysis

Link-analysis methods [23,22] define the importance of a schema element as a combination of the importance of the schema elements connected to it. Therefore, the more important the elements connected to a schema element are, the more important such schema element will be. In these methods the importance is

Table 3. Occurrence counting method applied to schema of Fig. 2

$r \in \mathcal{R}$	$\|conn(e_r)\|$	$\|rconn(e_r)\|$	$\boldsymbol{\mathcal{OC}(r)}$	$e \in \mathcal{E}$	$\|conn(e)\|$	$\|rconn(e)\|$	$\boldsymbol{\mathcal{OC}(e)}$
Grants	2	8	10	CreditCard	1	4	5
Has	2	13	15	Customer	3	9	12
Owns	2	7	9	Date	1	2	3
Serves	2	9	11	Restaurant	3	16	19
WorksIn	2	10	12	Table	2	10	12
Reservation	4	21	25	Waiter	2	9	11

shared through connections, changing from an element-centered philosophy to a more interconnected view of the importance.

Our approach adapts the link-analysis methods to compute the importance of entity types that are described in [24] to be used with associations. Concretely, we follow the same approach as the extended version of the EntityRank method (see Sec. 3.4 of [24]), which is based on Google's PageRank [4]. Each entity type in the schema is viewed as a state and each association between entity types as a bidirectional transition between them.

The link-analysis method we propose requires the reification of all the associations in the schema. Thus, we can compute their importance as in the case of entity types. Concretely, from a link-analysis perspective the importance of an entity type is the probability that a random surfer exploring the schema arrives at that entity type with random jumps (q component) or by navigation through associations ($1 - q$ component). Therefore, the resulting importance of the entity types in the schema with reifications $e \in \mathcal{E}^+$ correspond to the stationary probabilities of the Markov chain, given by:

$$\mathcal{LA}(e) = \frac{q}{|\mathcal{E}^+|} + (1-q)\left(\sum_{e' \in conn(e)} \frac{\mathcal{LA}(e')}{|conn(e')|} + \sum_{e'' \in rconn(e)} \frac{\mathcal{LA}(e'')}{|rconn(e'')|}\right)$$

Once we have the importance of all entity types $e \in \mathcal{E}^+$ in the schema with reifications, the next step is to obtain the importance of associations by analyzing the cases of those entity types $e \in \mathcal{E}_{\mathcal{R}}$ that are the reification of associations separately. Formally,

$$\mathcal{LA}(r) = \frac{\mathcal{LA}(e_r)}{\sum_{e \in \mathcal{E}_{\mathcal{R}}} \mathcal{LA}(e)}$$

where $\mathcal{LA}(r)$ is the relative importance of the association $r \in \mathcal{R}$, taking into account the importance $\mathcal{LA}(e_r)$ of its reification and the importance of the rest of reifications of associations in the schema $\mathcal{CS}^+$.

Table 4 shows the results of applying the link analysis method to associations and entity types of the example in Fig. 2. We forced that $\sum_{e \in \mathcal{E}} \mathcal{LA}(e) = 1$. Note that the link-analysis method discovers that the most important association is

Reservation but do not obtain big differences between the rest of associations. This situation occurs because of the small size of our example. According to our experience [24], this method should be used with schemas of larger sizes.

Table 4. Link analysis method applied to schema of Fig. 2

$r \in \mathcal{R}$	$\lvert conn(e_r)\rvert$	$\lvert rconn(e_r)\rvert$	$\boldsymbol{\mathcal{LA}(r)}$	$e \in \mathcal{E}$	$\lvert conn(e)\rvert$	$\lvert rconn(e)\rvert$	$\boldsymbol{\mathcal{LA}(e)}$
Grants	2	8	0.13	CreditCard	1	4	0.11
Has	2	13	0.16	Customer	3	9	0.21
Owns	2	7	0.15	Date	1	2	0.08
Serves	2	9	0.14	Restaurant	3	16	0.26
WorksIn	2	10	0.14	Table	2	10	0.17
Reservation	4	21	0.28	Waiter	2	9	0.17

4.3 Betweenness Centrality

Betweenness, in graph theory and network analysis [1], is a measure of the centrality of a vertex or an edge within a graph. It indicates the relative importance of such vertex/edge within the graph [2,3]. Since a conceptual schema can be seen as a graph with nodes (entity types) and edges (associations), it is possible to adapt the measure of node betweenness from graphs to compute the importance $\mathcal{BC}(r)$ of associations. Figure 3 shows a mini-example where *E5* is the most central (appears in a higher number of shortest paths) entity type, followed by *E3* and *E7*.

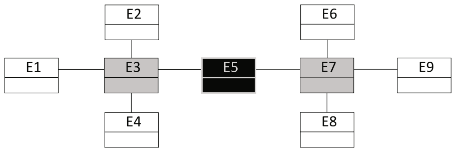


Fig. 3. Example of betweenness centrality of entity types in a conceptual schema

Basically, those associations that belong to more navigation paths in OCL and that their reifications are central in the schema are meant to be important. We firstly compute the betweenness of the entity types $e_r \in \mathcal{E}_{\mathcal{R}}$ that are reifications of associations $r \in \mathcal{R}$ by using the schema with reifications $\mathcal{CS}^+$. Formally,

$$\mathcal{BC}(e_r) = \sum_{e,e' \in \mathcal{E}^+} \frac{\mathcal{N}_{e,e'}(e_r)}{\mathcal{N}_{e,e'}},$$

where $\mathcal{N}_{e,e'}$ is the number of shortest paths between a pair of entity types $e, e' \in E^+$ in the schema with reifications traversing intrinsic associations $r \in \mathcal{R}^+$, and $\mathcal{N}_{e,e'}(e_r)$ is the number of those paths that go through e_r, which results from the reification of an association r.

Once we have the importance in the schema with reifications, the next step consists of obtaining the importance of associations of the original schema as in the case of the link-analysis method. Formally,

$$\mathcal{BC}(r) = \frac{\mathcal{BC}(e_r)}{\sum_{e \in \mathcal{E}_\mathcal{R}} \mathcal{BC}(e)}$$

To compute the shortest paths between entity types in the schema with reifications, we give a different length to each intrinsic association $r \in \mathcal{R}^+$ that appears in such schema, denoted by $\delta(r)$, according to its usage in OCL expressions. Concretely, we assign a shorter length for those intrinsic associations that are more navigated in order to favor their selection in the computation of shortest paths. To do so, we define in Table 5 a new measure $rconn(r)$ that computes the OCL navigations where an association r is traversed in the constraints $\mathcal{C}^+$.

Table 5. Definition of metrics for $r \in \mathcal{R}^+$

Notation	Definition
$rconn(r)$	$= \uplus_{c \in \mathcal{C}^+} \{\{e, e'\} \in links_{nav}(c) \mid association(\{e, e'\}) = r\}$
$\delta(r)$	$= \mathcal{M} - \lvert rconn(r) \rvert + 1$
$\mathcal{M}$	$= max_{r \in \mathcal{R}^+}(\lvert rconn(r) \rvert)$

Additionally, Table 5 includes the definition of the length $\delta(r)$ of an association of the schema with reifications. Note that $\mathcal{M}$ is the maximum number of navigations where an association $r \in \mathcal{R}^+$ is traversed. Therefore, those associations r types without occurrences in OCL expressions ($\lvert rconn(r) \rvert = 0$) will have a greater length (concretely, the greatest, $\delta_{max} = \mathcal{M} + 1$) than those with a big amount, and therefore their participation in shortest paths will be lower. On the contrary, the associations with a number of navigations closer to the maximum ($\lvert rconn(r) \rvert \approx \mathcal{M}$) will have a shorter length ($\delta \approx 1$), and will participate in shortest paths. Therefore, this approach uses the navigations through associations described in OCL expressions to compute the shortest paths in an importance-related way.

Table 6. Betweenness centrality method applied to schema of Fig. 2

$r \in \mathcal{R}$	$\mathcal{BC}(r)$	$\mathcal{BC}_\delta(r)$	$e \in \mathcal{E}$	$\mathcal{BC}(e)$	$\mathcal{BC}_\delta(e)$
Grants	0.14	0.07	CreditCard	0	0
Has	0.06	0.1	Customer	0.53	0.5
Owns	0.17	0.16	Date	0	0
Serves	0.13	0.16	Restaurant	0.25	0.19
WorksIn	0.07	0.05	Table	0.13	0.21
Reservation	0.43	0.46	Waiter	0.09	0.1

Table 6 shows the results of applying the betweenness centrality method to the example in Fig. 2. Note that $\mathcal{BC}_\delta$ takes into account the lengths $\delta(r)$ whereas $\mathcal{BC}$ does not. We forced that $\sum_{e \in \mathcal{E}} \mathcal{BE}(e) = 1$ and $\sum_{e \in \mathcal{E}} \mathcal{BE}_\delta(e) = 1$.

It is important to observe that without lengths the path from *Restaurant* to *Customer* through the association *Grants* is shorter than traversing the entity type *Table*. By contrast, since that second path is more navigated in OCL constraints, taking into account lengths we observe a significant reduction in the importance of *Grants* (0.14 to 0.07) and an increment in the importance of *Has* and *Table* (0.06 to 0.1 and 0.13 to 0.21, respectively). Therefore taking into account the lengths in associations according to their usage in OCL expressions is a more realistic approach to compute the importance.

5 Experimental Evaluation

We have implemented the three methods described in the previous section and we have evaluated them using five distinct case studies: the osCommerce [21], the EU-Rent [9], the UML2 metaschema [13], a fragment of the HL7 schemas [26], and the OpenCyc ontology [6]. For the case of the *betweenness centrality* method we used an existing implementation of the Brandes algorithm [7,3]. Table 7 summarizes the main characteristics of the five schemas.

Table 7. Schema elements of the case studies

	Entity Types	Associations	Constraints
osCommerce schema	346	183	457
EU-Rent schema	185	152	283
UML2 metaschema	293	377	188
HL7 schemas	2695	228	9
OpenCyc	2951	1385	0

In case of two or more entity types or associations get the same importance, our implementation is non-deterministic: it might rank first any of those. Some enhancements can be done to try to avoid ranking equally-important entity types or associations in a random manner, like prioritizing those with a higher amount of attributes or a higher amount of participations in OCL expressions (or any other measure) in case of ties. However, this does not have an impact to our experimentation. In the following, we summarize the main studies we have performed with the results of the application of the three methods.

5.1 Time Analysis

The first study we have made measures the execution time of each importance-computing method when applied to each of the five schemas. It is clear that a good method does not only require to achieve relevant results, but it also needs to present them in an acceptable time according to the user requirements. To find the time spent by our method it is only necessary to record the time lapse between the start of the method, and the receipt of the rankings of associations and entity types.

Figure 4 shows the execution time (in seconds) of all three methods in an Intel Core 2 Duo 3GHz processor with 4GB of DDR2 RAM. According to the results,

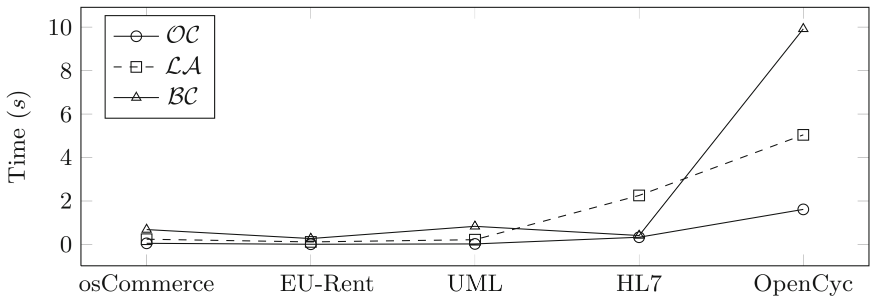


Fig. 4. Execution time between methods for each schema

the *occurrence-counting* method ($\mathcal{OC}$) is the fastest method for all the schemas, because of its simplicity with respect to the other two methods. On the contrary, the *betweenness-centrality* method ($\mathcal{BC}$) is the slowest one due to its bottleneck on computing the shortest paths between schema elements. Despite that, the $\mathcal{BC}$ method performs better than the *link-analysis* method ($\mathcal{LA}$) in the case of the HL7 due to the reduced number of associations it contains (in comparison to the number of entity types), which dramatically reduces the previous bottleneck of the shortest-paths computing process.

Consequently, if the method is intended to be used for determining the importance of associations and entity types in a context where the target schema is rapidly evolving through changes, it is better to select a faster method like the $\mathcal{OC}$, or even the $\mathcal{LA}$ (taking into account the size of the schema). Otherwise, if the context is static and the schema does not change, the method to select may be any of the three (the importance could be pre-calculated without runtime consumption).

5.2 Correlation between Methods

We apply the three methods to each of the five schemas. Each method can compute the importance of associations and entity types, and therefore produces two different rankings of schema elements: the one of entity types and the one of associations. Our research aims to know which methods give similar results, in order to select the simpler method in any case. Thus, we study the correlation between methods by analyzing the correlation of the rankings they produce.

Figure 5 shows, for each pair of methods, the results obtained in the correlation analysis for the rankings of importance of associations. Our aim is to know whether it is possible to compare the results of the importance methods and to search for a common behavior. We can observe that the most correlated methods to compute the importance of associations are the *occurrence-counting* and *link-analysis* methods, although there exists a certain variability in the results, denoted by the fact that their correlation in larger schemas tend to decrease.

Figure 6 shows, for each pair of methods, the results obtained in the correlation analysis for the rankings of importance of entity types. In this case, although

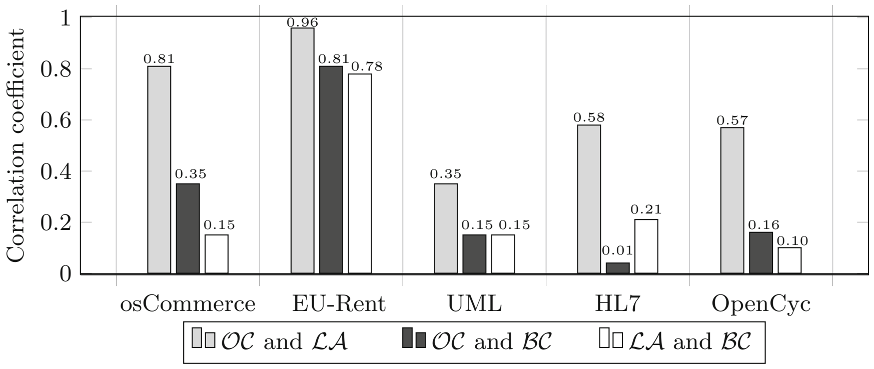


Fig. 5. Correlation between methods for each schema in the case of the importance of associations

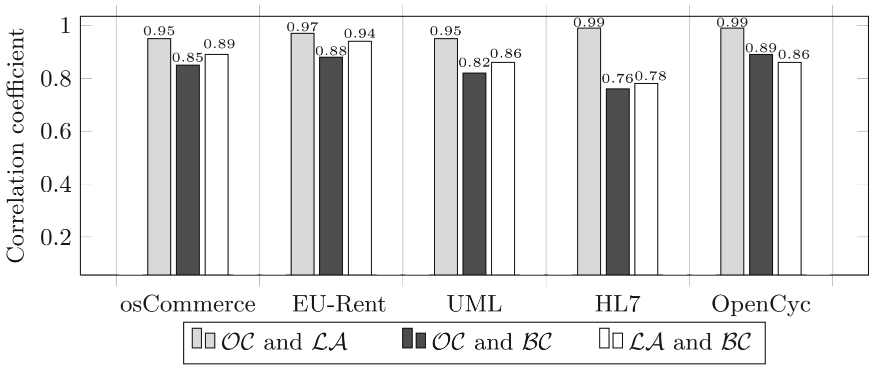


Fig. 6. Correlation between methods for each schema in the case of the importance of entity types

the pair of methods with a higher correlation are the same than in the case of associations ($\mathcal{OC}$ and $\mathcal{LA}$), all the methods produce similar rankings. It means that the three methods of importance are more consistent when applied to entity ranks than when applied to associations.

The fact that the pair of methods $\mathcal{OC}$ and $\mathcal{LA}$ produce more similar results between them than with the *betweenness-centrality* method ($\mathcal{BC}$) is due to the different approach each method follows. On one hand, $\mathcal{OC}$ and $\mathcal{LA}$ compute the importance of schema elements by counting the characteristics they have (and the characteristics of the elements connected to them, in $\mathcal{LA}$) in the schema. On the opposite, $\mathcal{BC}$ computes the importance by analyzing the topology and structure of the schema measuring the shortest paths between elements and their participation in those shortest paths.

Therefore, to select a method or another depends on the approach the user wants to follow to compute the importance of associations. An approach closer to counting the number of characteristics the associations have, will choose $\mathcal{OC}$

or $\mathcal{LA}$, while an approach closer to select those associations that are more central in the schema will choose $\mathcal{BC}$. For the case of entity types, as shown in Fig. 6, the selection of a method or another has a lower impact in the obtained results, because of the greater correlation between them.

5.3 Impact of the Reifications

We have analyzed the results of the three methods when applied to compute the importance of the entity types of each of the five test schemas in their original form ($\mathcal{CS}$) and after their reification ($\mathcal{CS}^+$). The resulting correlation between $\mathcal{CS}$ and $\mathcal{CS}^+$ is an indicator of the impact that reifications have in the importance of entity types.

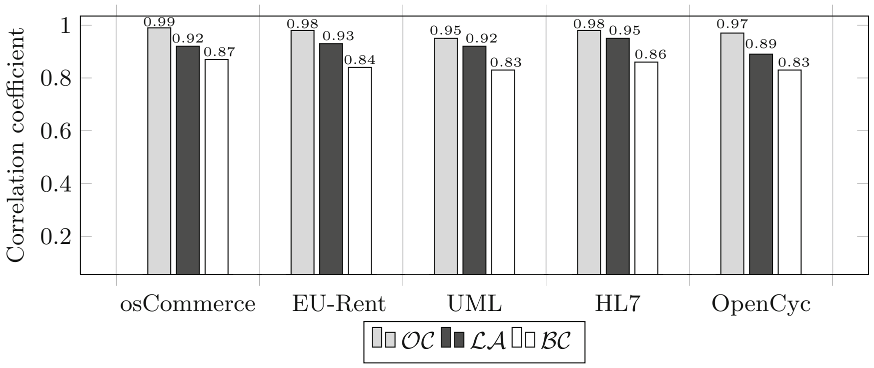


Fig. 7. Correlation between results before and after reifications for the case of the importance of entity types

Figure 7 shows that the transformation of the associations into entity types through the reification process has an impact in the importance of entity types by introducing changes in the resulting rankings. Those changes are mainly produced by the addition of implicit associations after reifications to maintain the connections. However, the correlation between the rankings of importance of entity types in $\mathcal{CS}$ and $\mathcal{CS}^+$ is close to 1, which means that the impact of reifications is minimal. Therefore, although $\mathcal{CS}^+$ changes with respect to $\mathcal{CS}$, reifying associations produces a low impact on the computed importance of entity types.

6 Conclusions and Further Work

The visualization and the understanding of large conceptual schemas require the use of specific methods. These methods generate indexed, clustered, summarized or focused schemas that are easier to visualize and understand. Almost all of these methods require computing the importance of each element in the schema but, up to now, only the importance of entity types has been studied in the literature.

The computed importance induces an ordering of the elements, which plays a key role in the steps and result of the methods that deals with large schemas. We have proposed three methods to compute the importance of associations, and also entity types. The methods we describe are based on *occurrence-counting* ($\mathcal{OC}$), *link-analysis* ($\mathcal{LA}$), and *betweenness-centrality* ($\mathcal{BC}$). Our approach transforms the schema by reifying the associations into entity types. As a result, we use existing importance-computing methods from the literature with minor modifications to be able to work with associations.

We have implemented the three methods in a prototype tool and we have experimented them with five large real-world conceptual schemas. The results we obtained indicate that the quickest method is the $\mathcal{OC}$, which must be selected if the interaction context is dynamic and the user wants real-time feedback. Furthermore, we observe that the computed rankings of importance for entity types have a greater similarity independently of the selected method which indicates that all three methods are indistinguishable in that aspect. Conversely, for the case of the rankings of associations, the selection of a method or another has an impact in the obtained results.

The combination of the importance of entity and associations our methods compute can be applied to several techniques to deal with large conceptual schemas in order to reduce the effort a non-expert user must do to understand the knowledge within the schema. An example is the construction of reduced schema summaries with the top of both importance rankings to give a simple view of the schema contents. We plan to continue our work in that direction.

Acknowledgements. This work has been partly supported by the Ministerio de Ciencia y Tecnologia and FEDER under project TIN2008-00444/TIN, Grupo Consolidado, and by Universitat Politècnica de Catalunya under FPI-UPC program.

References

1. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.: Complex networks: Structure and dynamics. Physics Reports 424(4-5), 175–308 (2006)
2. Brandes, U.: A faster algorithm for betweenness centrality. Journal of Mathematical Sociology 25, 163–177 (2001)
3. Brandes, U.: On variants of shortest-path betweenness centrality and their generic computation. Social Networks 30(2), 136–145 (2008)
4. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems 30(1-7), 107–117 (1998), Proceedings of the 7th International World Wide Web Conference
5. Castano, S., De Antonellis, V., Fugini, M.G., Pernici, B.: Conceptual schema analysis: techniques and applications. ACM Transactions on Database Systems 23(3), 286–333 (1998)
6. Conesa, J., Storey, V.C., Sugumaran, V.: Usability of upper level ontologies: The case of ResearchCyc. Data & Knowledge Engineering 69(4), 343–356 (2010)
7. Dutot, A., Guinand, F., Olivier, D., Pigné, Y.: Graphstream: A tool for bridging the gap between complex systems and dynamic graphs. EPNACS: Emergent Properties in Natural and Artificial Complex Systems (2007), `http://graphstream-project.org`
8. Egyed, A.: Automated abstraction of class diagrams. ACM Transactions on Software Engineering and Methodology (TOSEM) 11(4), 449–491 (2002)

9. Frias, L., Queralt, A., Olivé, A.: EU-Rent Car Rentals Specification. LSI Research Report. Tech. rep., LSI-03-59-R (2003), http://www.lsi.upc.edu/dept/techreps/techreps.html
10. Jaeschke, P., Oberweis, A., Stucky, W.: Extending ER Model Clustering by Relationship Clustering. In: Elmasri, R.A., Kouramajian, V., Thalheim, B. (eds.) ER 1993. LNCS, vol. 823, pp. 451–462. Springer, Heidelberg (1994)
11. Lindland, O.I., Sindre, G., Sølvberg, A.: Understanding quality in conceptual modeling. IEEE Software 11(2), 42–49 (1994)
12. Moody, D.L., Flitman, A.: A Methodology for Clustering Entity Relationship Models - A Human Information Processing Approach. In: Akoka, J., Bouzeghoub, M., Comyn-Wattiau, I., Métais, E. (eds.) ER 1999. LNCS, vol. 1728, pp. 114–130. Springer, Heidelberg (1999)
13. Object Management Group: Unified Modeling Language (UML) Superstructure Specification, version 2.2 (February 2009), http://www.omg.org/spec/UML/2.2/
14. Object Management Group: Object Constraint Language Specification (OCL), version 2.0 (February 2010), http://www.omg.org/spec/OCL/2.2/
15. Olivé, A., Cabot, J.: A research agenda for conceptual schema-centric development. In: Conceptual Modelling in Information Systems Engineering, pp. 319–334 (2007)
16. Olivé, A.: Conceptual Modeling of Information Systems. Springer, Heidelberg (2007)
17. Olivé, A., Raventós, R.: Modeling events as entities in object-oriented conceptual modeling languages. Data & Knowledge Engineering 58(3), 243–262 (2006)
18. Schewe, K.-D., Thalheim, B.: Conceptual modelling of web information systems. Data Knowl. Eng. 54(2), 147–188 (2005)
19. Schmidt, P., Thalheim, B.: Management of UML Clusters. In: Abrial, J.-R., Glässer, U. (eds.) Borger Festschrift. LNCS, vol. 5115, pp. 111–129. Springer, Heidelberg (2009)
20. Thalheim, B.: Entity-relationship modeling: foundations of database technology. Springer, Heidelberg (2000)
21. Tort, A., Olivé, A.: The osCommerce Conceptual Schema. Universitat Politècnica de Catalunya (2007), http://guifre.lsi.upc.edu/OSCommerce.pdf
22. Tzitzikas, Y., Kotzinos, D., Theoharis, Y.: On ranking RDF schema elements (and its application in visualization). Journal of Universal Computer Science 13(12), 1854–1880 (2007)
23. Tzitzikas, Y., Hainaut, J.-L.: How to Tame a Very Large ER Diagram (Using Link Analysis and Force-Directed Drawing Algorithms). In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, pp. 144–159. Springer, Heidelberg (2005)
24. Villegas, A., Olivé, A.: Extending the methods for computing the importance of entity types in large conceptual schemas. Journal of Universal Computer Science (J.UCS) 16(20), 3138–3162 (2010)
25. Villegas, A., Olivé, A.: A Method for Filtering Large Conceptual Schemas. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, pp. 247–260. Springer, Heidelberg (2010)
26. Villegas, A., Olive, A., Vilalta, J.: Improving the usability of HL7 information models by automatic filtering. In: IEEE 6th World Congress on Services, pp. 16–23 (2010)
27. Yang, X., Procopiuc, C.M., Srivastava, D.: Summarizing relational databases. In: 35th Intl. Conf. on Very Large Data Bases, VLDB 2009, pp. 634–645 (2009)
28. Yu, C., Jagadish, H.V.: Schema summarization. In: 32nd Intl. Conf. on Very Large Data Bases, VLDB 2006, pp. 319–330 (2006)

Conceptual Modeling of Human Genome: Integration Challenges

Oscar Pastor, Juan Carlos Casamayor, Matilde Celma, Laura Mota, M. Ángeles Pastor, and Ana M. Levin

Centro de Investigación en Métodos de Producción de Software (PROS)
Departamento de Sistemas Informáticos y Computación,
Universitat Politècnica de València, Camino de Vera S/N. 46022, Valencia, Spain
`opastor@pros.upv.es`

Abstract. While Information Systems (IS) principles have been successfully applied to the design, implementation and management of a diverse set of domains, the Bioinformatics domain in general and the Genomic one in particular, often lacks a rigorous IS background, based on elaborating a precise Conceptual Model where the relevant concepts of the domain were properly defined. On the contrary, current genomic data repositories focus on the solution space in the form of diverse, ad-hoc databases that use to be hard to manage, evolve and intercommunicate. Conceptual Modeling as a central strategy is then far from the current biological data source ontologies that are heterogeneous, imprecise and too often even inconsistent when compared among them. To solve this problem, a concrete Conceptual Schema for the Human Genome (CSHG) is introduced in its latest version on this chapter. With a holistic perspective, the CSHG focuses on the different genomic views that must be integrated and emphasizes the value of the approach in order to deal appropriately the challenge of correctly interpreting the human genome.

Keywords: Conceptual Modeling, Information Systems in Bioinformatics, Semantic Integration, Human Genome.

1 Introduction

Nowadays, the importance of Conceptual Modeling (CM) is widely accepted in the Information Systems (IS) community. If designing and implementing a correct information system is the goal, a conceptual schema becomes the essential software artifact [1]. Conceptual modeling is the appropriate key activity to understand the relevant concepts of the domain, to represent them accordingly to a selected notation, and to transform their representation accurately into the subsequent software components that will conform the final system.

During the last decades, a lot of knowledge and experience has been accumulated in the IS area, and a huge number of applications –mainly database-oriented applications– covering practically any human working domain, have put IS on the top of the technological wave. Facing any domain, understanding it through the corresponding conceptual modeling framework and defining its conceptual schema have become the central part of any software production process. Ontologically speaking, conceptualization is essential for this mandatory understanding process, and a conceptual schema is seen as the ontological representation of the relevant knowledge of a selected universe of discourse. A set of modeling elements or conceptual constructs provides the basis to determine the breaks that will build the conceptual schema. A good example of this conceptual characterization can be found in the FRISCO proposal [2], where organizational systems concepts were introduced, with a sound, ontological background that makes possible to elaborate conceptual modeling-based software processes oriented to design and implement this type of IS, as the OO-Method presented in [3].

Having the conceptual schema that corresponds to a target domain, modern model engineering strategies (Model-Driven Development, Extreme Non-Programming, Conceptual Schema-Centric Software Development or MDA-based CASE tools) provide a set of diverse technologies that allow to transform models, from the early requirements models to the final software models. In these technologies, conceptual models always play a role as the essential software artifact on which the full software production process pivots.

We find IS-based applications in almost every relevant organizational domain (universities, hospitals, banking, any type of shopping centers, companies, …) covering a huge set of different human activities (accounting, selling, purchasing, stock control, staff management, …). Therefore, it is interesting to analyze how this IS background is affecting the development of bioinformatics, a modern and challenging domain that nowadays concentrates a lot of interest and where IS principles –especially a sound Conceptual Modeling-based perspective- are not so clearly applied.

Data management –an essential feature of any IS-based application domain– is a very big issue in the bioinformatics area, especially in the context of Genomics, and more particularly, in the human genome domain. Literally, tons of genomic data are produced every day and are stored in very different sources, intended to provide an adequate answer to one of the most outstanding challenges of the XXIst century: understanding the human genome. The need for emerging technologies to face this proliferation of data resources is a problem that has already been pointed out by different authors [4, 5]. This need presents an even bigger challenge for ‘omics” researchers who need to automate large-scale data aggregation across many different sites. Additionally, the continuous need of collecting all the phenotypically interesting variants of the human genome requires the use of new strategies of data storage and reference [6].

But, while this huge amount of data is feeding a large number of biological databases, it is highly remarkable the absence of a formal IS approach to understand this domain, and the representation of the current knowledge in a precise conceptual schema that could be called “Conceptual Schema of Human Genome”. This schema should provide a reliable basis to fix, discuss and understand, without ambiguity, which are the precise biological concepts that conform the current state of the knowledge associated to the human genome. Additionally, as new information is continuously generated in this domain, a conceptual-schema centric approach is also intended to provide a more efficient conceptual framework much better prepared to face evolution of concepts than other approaches that are strictly centered on concrete database schemas.

There are many biological databases and domain ontologies covering different genomic perspectives (Gene Ontology [7,8], Entrez-Gene [9], Ensembl [10,11], GeneBank [12] or HGMD[13,14] for understanding mutations; OMIM [15,16] for representing phenotypes related to human illnesses; KEGG [17,18], UniProt [19,20], InterPro [21,22], for understanding proteins and their interactions; Reactome [23,24] for understanding pathways, and many much more). But when we look for a common conceptual schema to have a holistic, unified conceptual view of all this knowledge, it is difficult to find a clear response. If the key question to be answered in this context is the definition of a precise set of semantic mappings between genotype (the "source" genetic code) and phenotype (its external, perceivable characteristics), it is clear that such a unified conceptual view is a strong need. This integrated conceptual schema should become the cornerstone of any well-founded biological exploitation of reliable data, especially considering the high rate of modification that affects the involved domain of knowledge.

Following the research direction introduced in [25], our work aims to provide an answer to this challenge, under the belief that only by having a well-defined conceptual schema of the human genome, it would be possible to put the required order in the current chaotic management of genomic data. We refer to this vast set of data spread out in many different data sources, with different formats, too often inconsistent, redundant and with a doubtful reliability in terms of data validity as the "genome data chaos". An additional problem of this context is that day after day new data become available and some of the existing one change. The continuous advances in DNA sequencing strategies and their cost reduction accelerate dramatically this phenomenon. If we consider just the human genome case, we talk about 30.000 genes whose individual behavior, in terms of detailed functionality, is in most of the cases still ignored and whose precise interactions open a huge field of research still to be explored. It seems evident that we face the challenge of modern times: managing adequately this immense set of data, in constant evolution, and with underlying concepts whose definition can often be considered still work in progress. This challenge has an evident IS-oriented "taste", and it can only be correctly approached if we take careful advantage of the good practices accumulated during years in the IS arena.

In this context, this chapter introduces a complete conceptual schema of the human genome, intended to provide a conceptual framework to develop a correct Genomic Information System –GIS-. By GIS, we mean to look at the Genome as a concrete Information System where the relevant concepts are those of the Molecular Biology, intended to understand how to link genome structure with phenotype expression. This is the main objective of this work: to provide the subsequent Conceptual Schema that should make possible to understand and interpret the genome, including its different perspectives (genotype, phenotype, pathways and any other relevant information). Under the quoted, current chaos in the genomic data management, the proposed conceptual schema is a main contribution in the bioinformatics area, constituting a well-defined link between Information Systems principles and Bioinformatics works oriented to understand the language of life represented by the genome.

One important aspect of the genomic domain is how dynamic the current knowledge about it is. Day after day, new information is generated as the result of the huge amount of research that is being developed. This evolution affects directly the completeness and correctness of a Conceptual Schema as the one that we want to create. But this is at the same time a main benefit of following a Conceptual Modeling-centric approach. Using the Conceptual Schema as the central repository of current, relevant genomic information, genomic data evolution can be much better managed. The evolution of our work is –on the one side- discovering new genomic information that must be incorporated into the model, while on the other side conceptual updates are precisely located, understood and traced to the corresponding data sources –either databases or data ontologies-.

Another important advantage is to facilitate data integration from heterogeneous data sources: only having a precise Conceptual Schema as the central, essential component, pieces of information stored in diverse data sources can be adequately interconnected.

However, we want to emphasize that this work should not be seen as just an attractive exercise of conceptual modeling in a fashionable domain. The joint work performed by information system experts and biology experts provides a very interesting example of how conceptual modeling is a powerful tool to determine precisely the semantics of basic genome-related concepts that sometimes are not as clear as one could expect. The definition of the proposed conceptual schema has lead to interesting discussions where often basic concepts have been strongly discussed raising successive conceptual schema versions, and even restructuring initial knowledge when new insights in the definition of those concepts were introduced. To fix the meaning of gene, alleles, mutations, transcripts, splicing or SNPs among many others, originated rich discussions, and this is from our perspective a second, major contribution of this work: to show how the use of a sound conceptual modeling framework leads to a much powerful context to make feasible and reliable the major value of our expected result, that is the adequate interpretation and understanding of the human genome.

According to these objectives, the structure of the presented chapter is concise: after this introduction where the problem is stated and our basic objectives are explained, the most significant related work developed in this context is discussed in section 2. The proposed conceptual model of the human genome is then developed in section 3, showing its structure in five different views that are properly integrated. These five views are the structural view –where the main basic genome-related concepts are introduced-, the transcription view –where the concepts related to how the source DNA is processed to synthesize proteins are presented-, the variation view –intended to show how changes in the genome affect the expected normal behavior-, the pathway view –that adds the cellular metabolism principles that are required to understand relationships between genotype and phenotype- and the source & bibliography view –oriented to assure that the source of reliable information is properly documented-. The adequate integration of the information involved in these views provided the full picture of a complete Conceptual Schema ready to be used and managed according to those well-founded IS principles needed to assure the successful manipulation of the genomic data. Conclusion and references close the work.

2 Related Work

It is true that many different databases related to the storage of different types of genomic-oriented information can be found in the biological domain. We have quoted some of the above in the introduction. But while it is usual to see diverse biological databases used to manage different types of genome data, it is not so normal to have an associated sound and rigorous conceptual schema background behind them. In the Bioinformatics domain they are often referred as data sources, data repositories or ontologies. The use of the "ontology" term is in that context somewhat controversial, as those data sources often represent more a glossary of genomic terms, than a true, shared conceptualization of the domain as an ontology is supposed to do. We would not say that data sources as RefSeq, Gene Ontology, Entrez Gene, Ensembl… are precise ontologies in the "pure", philosophical notion of ontology. Instead, they can be seen closer to the notion of databases, centered around a very concrete database schema, without a precise conceptual schema associated to its definition, and totally concentrated on the solution space perspective instead of on the problem space. As we look at a conceptual schema as a concrete representation of an ontology, the lack of such a conceptual background in the most widely-known and widely-used genomic databases constitutes a problem that our work faces directly. The Conceptual Schema of the Human Genome that we present in this chapter would provide the required "conceptual structure" needed to store each piece of genomic data in the right place, independently of the selected origin of data.

This way has been previously explored by a few authors, that we consider pioneers of our proposal. Paton et al. were in [26, 28, 29] the first ones introducing the idea of modeling the genome, and they introduced a first set of data models intended to achieve this goal. In their proposal a collection of data models for genomic data is presented. These models describe elements involved in transcriptional and translational processes as well as the variant effects generated by them. Their work could be said to be preliminary, and it had no clear continuation, but in some way what we present here is based on the shared principle of using conceptual modeling as the key artifact for managing genome data. The results that we present can then be seen as a major extension and conceptual enrichment of those ideas.

Moreover Ram has successfully applied in [27] conceptual modeling principles in the context of the protein notion. More concretely, in her work it is shown how 3D protein structure search and comparison can be facilitated through conceptual modeling, not only enabling to predict unknown structures, but also revealing distant evolutionary relationships that are otherwise undetectable, and perhaps suggesting unsuspected functional properties. Even if the considered domain is more restricted, the work demonstrates that conceptual modeling provides the adequate background to manage data more effectively. That paper concentrates on the 3D protein structure, while in this chapter our goal is to introduce a holistic perspective of the human genome data model; in some way, the work presented by Ram could be embedded in the global perspective that our proposal wants to provide.

We find these conceptual modeling-based experiences in other genomic domains different from the human genome one. Some other attempts to model more viable genomes have been reported. An interesting experience is provided by the e-Fungi initiative [30, 31], where a systematic comparative analysis of fungal genomes is supported. The e-Fungi database integrates a variety of data for more than 30 fungal genomes and it provides fungal biologists with a powerful resource for comparative studies of a large range of fungal genomes. This work is developed in a different domain -the fungi genome instead of the more complex human genome- but it shows a clear path of results exploitation that could be perfectly projected to our Human Genome modeling effort.

In order to close this related work analysis, we want to mention our previous, first attempts to build a conceptual schema for the human genome, published in [32, 33]. These very first proposals constitute the starting point to the conceptual schema presented in this chapter. While we have been increasing the perspective of our conceptual modeling-based approach to the genomic domain, these works present initial and partial views of the whole picture, that help to understand how the current state has been achieved. The conceptual schema views reported in these works have been largely corrected and extended. Even if not all the components were present as we do in this chapter, the evolution followed by the conceptual representation of the genomic knowledge, and how difficult is to synchronize the discourse and the communication between IS experts and biologists, becomes evident when following the subsequent schema evolution reports. We are conscious about the difficulty of this "conceptual trip". We know that what appears to be correct today according to the current genomic knowledge, may change tomorrow and may need to be updated. But we are convinced that our conceptual model-driven perspective is the right choice to manage this domain that is in continuous evolution.

It is especially relevant in this chapter the presentation of the five views that we introduce next, including metabolic pathways and data provenance notions that are essential to deal with all the genomic information. We refer to the "integration challenge" problem as how well all of them together provide the required full perspective of the genome ontological background.

These works are a few of the existing examples about the use of conceptual modeling in bioinformatics applications. They can be used to prove that conceptual modeling is an effective approach to help to improve biologic research. It is our belief that the work described in this chapter is an important contribution to the global understanding of the human genome, because only having a conceptual schema to characterize it, it will be possible to store the right contents, to manage them efficiently, and to understand the precise relationships existing between phenotype (external manifestation of human properties) and genotype (their corresponding genomic code).

3 A Conceptual Schema for the Human Genome

In this section, we present the current version of the Conceptual Schema of the Human Genome (CSHG).

To simplify the presentation of the scheme, five conceptual views are considered: the structural view, the transcription view, the variation view, the pathway view and the data source and bibliography view. The joins between these views are pointed out by shadowed boxes.

It is important to remark that this presentation is built over the experience acquired in our joint work with biologists and bioinformaticians during the last couple of years, where previous attempts to construct the conceptual schema have been restructured and extended. Different, partial versions have been presented in the past, and in this chapter we introduce the latest one, where the very important metabolic pathway perspective is included, and where a set of relevant, organizational changes are introduced as a result of a more detailed knowledge of the genomic domain. With these changes, we claim to provide the most complete conceptual schema for the human genome that –up to what we know– currently exists. Understanding all the pieces of genomic information can be a hard task that we try to alleviate with additional explanations. In any case, we invite the reader to appreciate the value of having the most relevant concepts needed to understand the human genome all together collected and represented in just one whole conceptual schema (CSHG).

3.1 Structural View

A genome is defined as all the genomic information carried by an organism, in this case a human being. This information is codified in 3 billion base pairs that constitute our DNA. The DNA structure is a double helix of complementary and anti-parallel strands, where the nucleotides (A, C, G, T) stand inside and the sugar-phosphate backbone outside. Due to complementary base pairing, A always binds together with T and C with G. Considering the enormous size of the DNA molecule, the information is distributed in 23 pairs of chromosomes. Inside them we can find protein coding genes (around 23.000) but also non-coding DNA, regulatory sequences or introns [34]. To model its internal structure, some significant chromosomal fragments have to be considered (Figure 1).

Although our current interest is just on the human genome, the conceptual model intends to cope also with genomes from different species. To represent which species one chromosome belongs to, a class *Species* is included in the schema. The relevant properties of this class are the following: *scientific_name, common_name, ncbi_taxon_id* (identification given to the species by the NCBI organization), *assembly* (identification of the version used), *date_assembly* (version creation date) and *source* (where the DNA sequences used as references are obtained from).

The main class in this view is the *Chromosome* class. It is important to remark that the presented structural view is centered on the concept of chromosome and the different types of chromosome elements that have distinctive properties, instead of focusing strictly on the more conventional notion of gene that can be found in [35, 36] . With this modeling strategy, we go one step forward in the evolution of the gene concept –see [37] for a complete review of how the term has being evolved until now-, proposing an alternative perspective that we find to be more accurate to define adequately the basic, structural genome terms. A chromosome is an organized structure of DNA and protein that is found in cells. It is a single piece of DNA where genes, regulatory elements and other nucleotide sequences are located. The chromosome has a *name* (chromosome identification at the source where the reference sequence is obtained), a *sequence* (reference sequence) and a *lenght* (how many nucleotides the sequence has). There are two important things to emphasize in this class, the first one is that different genome versions will be stored in different databases, and the second one is that the reference sequence does not correspond to any particular individual.

To describe other features of a chromosome, there are two classes in the view: *hotspot* and *cytoband*. To understand those concepts we have to understand Meiosis and how cells assure that all the genetic material is present every time they divide. Meiosis is the process where sexual cells from our body are formed. During this process, chromosome pairs align on the cell´s equator and are pulled apart, so only one copy of all of the chromosome pairs ends up at opposite sides of the cell. To ensure a proper alignment, the chromosomes exchange certain sections of genes. This is also called genetic recombination and is an essential part of sexual reproduction, since it ensures greater genetic diversity. Recombination is an extraordinarily precise process where segments exchanged have to be exactly the same length, to avoid "frame-shift" mutations. This is a rather important phenomenon, since wrong recombination could produce misalignment that can result in both copies of a chromosome ending up in one cell. When this happens, the cells die or suffer from severe genetic problems, such as Down syndrome [38].

Those chromosome zones where recombination occurs most frequently are called hotspots (for a more detailed explanation see [39]). Consequently, the *hotspot* class represents information about the points in the sequence where this process happens. It has two attributes: *hotspot_id*[1] and *position*. The *cytoband* class (or cytogenetic band class) refers to stored information about the subregions of a chromosome that becomes microscopically visible after a staining during a specific cell cycle phase (Metaphase). This is important since it is reproducible and allows identifying each chromosome pair, like a barcode, but also permits physical localization of genes on the chromosome. It is even demonstrated that there are patterns of mutational input related to chromosome bands [40]. Our *cytoband* class has four attributes, *name* (always in the same format, following the naming rules where "p" or "q" describes the arm of the chromosome followed by a number with one, two or three digits separated with dots, depending on the resolution used), *score* (staining intensity which can take five different values proportional to the presence of A and T nucleotides), *start_position* (initial position in the chromosome reference sequence) and *end_position* (final position in the chromosome reference sequence).

The *chromosome_element* class represents the information about relevant chromosomal fragments. It has four attributes: *chromosome_element_id*, *start_position* (initial fragment position in the reference sequence), *end_position* (final fragment position in the reference sequence) and *strand* (the strand of the double helix where the fragment is found).

[1] The attributes ending with _id denote internal identifiers.

Chromosome elements are classified in three types: *transcribable_element* (chromosome element that is transcribed), *regulatory_element* (chromosome element with regulatory function) and *conserved_region* (chromosome element that occurs in many species). The conserved regions usually tend to be non-coding regions and the attribute *score* represents the region´s conservation degree (or a statistical value indicating a probability or an output value of a formula); it is a real number, the larger, the more conserved.

Transcribable elements can be of two types: *gene* and *exon*. A gene is a DNA region with regulatory elements (promoters, activators, etc.) that control the transcription process. The *gene* class has five attributes: *ensemble_gene* (gene name given by Ensembl[2]), *description*, *biotype* (possible values: snRNA (Small nuclear ribonucleic acid, a class of RNA involved in processes such as splicing or telomere maintenance), protein coding (RNA that will produce proteins), miRNA (MicroRNA, short RNA molecule involved in gene silencing, etc.), *status* (possible values: deprecated, new, etc.) and *gc_percentage* (percentage of GC base pairs in the element). An exon is a gene fragment that is the basic unit of the transcripts.

If a gene produces transcription factors it will be included in the *tf* class. The attribute *cons_seq* in this class represents the consensus sequence.

Regulatory elements are DNA regions that affect gene expression either enhancing or repressing it. They can work on two different levels, directly regulating the expression of a gene (*gene_regulator)* or indirectly regulating its transcript (*transcript_regulator).*

A gene regulator can be a *tfbs* (transcription factor binding site), a *cpg_island* or a *triplex.* A *tfbs* is a DNA region where transcription factors bind producing an effect on gene transcription (activation or repression). The class where these elements are represented has five attributes: *name, type, description, score* (degree of similarity between the consensus sequence and the *tbfs*) and *cons_seq* (consensus sequence which binds the *tbfs*). A *cpg_island* is a set of CG repetitions close to the promoter that are targets for methylation, which is another way to alter gene expression; the attribute *cg_percentage* is the percent of GC nucleotides in the element. Finally, a *triplex* is a DNA region a where the structure is a triple helix.

Moreover, a transcript regulator can be of two types: *mirna_target* (a target of a microRNA element) and *splicing_regulator* with two properties: *type* (possible values: enhancer, silencer) and *regulated_element* (intron or exon).

[2] The Ensembl Project produces genome databases for vertebrates and other eukaryotic species, and makes this information freely available online.

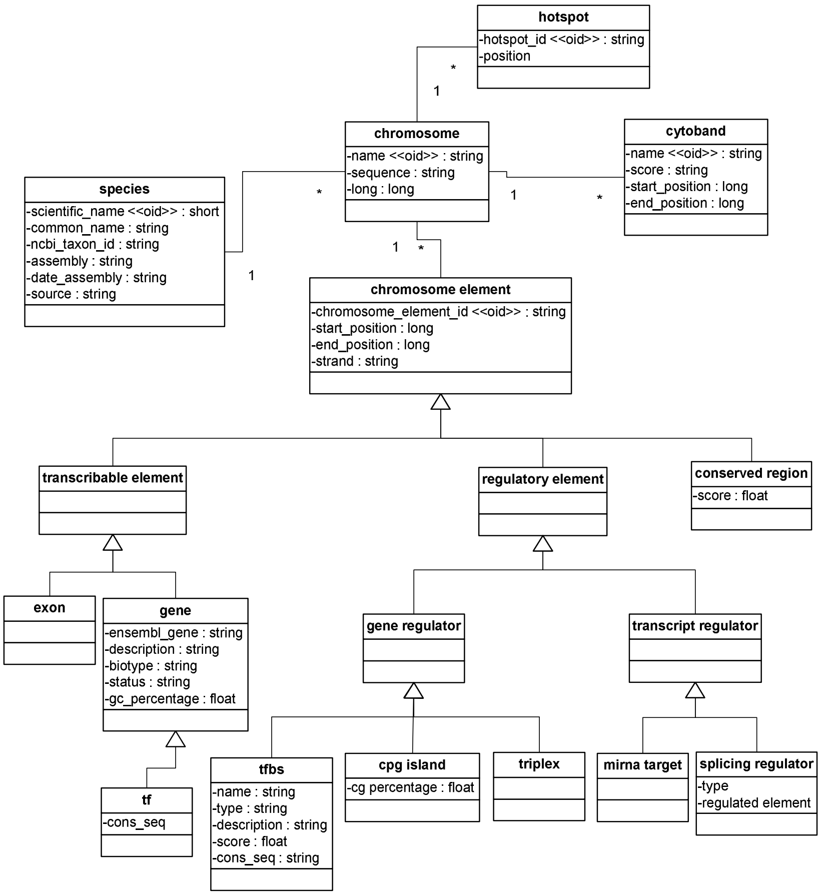


Fig. 1. Structural view

3.2 Transcription View

A significant number of genes express their functional effect through the production of proteins. This process begins with the formation of an RNA molecule whose sequence is complementary to the genomic DNA sequence. The transcription view shows the components and concepts related to protein synthesis. The DNA sequence that is transcribed into an RNA molecule encodes at least one protein. If the transcribed gene encodes for a protein, the transcription result is a messenger RNA (mRNA), which will then be used to create a protein via the translation process.

After transcription, an RNA modification called splicing takes place. During the splicing, introns are removed and exons are joined. In many cases, the splicing process can affect exon composition of the same messenger RNA. This phenomenon is called alternative splicing and it is the reason why one gene encodes for more than one protein. Alternative splicing can occur in many ways. Exons can be extended or skipped, or introns can be retained. A more detailed description of these biological processes can be found in [41].

In the Transcription view (Figure 2), the *Transcript* class represents the different transcripts that can be produced from genes. Each transcript may have different functions, represented in the *biotype* attribute, whose value can be ‘Protein Coding’, ‘tRNA’(for Transfer RNA, the molecule that “reads” the nucleotide sequence and traduce it to aminoacids), ‘rRNA’(for Ribosomal RNA, a structural RNA that forms the ribosomes, organelles involved in protein synthesis), ‘miRNA’, ‘siRNA’ (for small interference RNA, small double strand RNA involved in the process of RNA interference that modulates gene expression), ‘piRNA’ (for Piwi-interacting RNA, small RNA molecules involved in silencing of retrotransposons), ‘Antisense’(RNA molecule that is complementary to mRNA, this molecule sticks to it and represses its expression), ‘Long noncoding’ (RNAs that do not code for proteins but with several other functions, not as relevant as the small ones), ‘Riboswitch’ (part of a mRNA involved on its own regulation), ‘snRNA’, ‘snoRNA’ (small nucleolar RNAs or guide RNAs, involved in modifications of other functional RNAs), ‘mitochondrial’ (RNA produced by DNA that is inside mitochondrias, organelles that have its own genetic material), or others.

The relationship between *Transcript* class and *Exon* class represents the information about the exons combined in a transcript.

The *Protein coding* class is a specialization of the *Transcript* class representing the first of the biotypes cited above. It is necessary because the translation start position and the translation end position have to be stored.

From a *Protein coding* transcript, many different proteins can be synthesized. The *Protein* class stands for proteins; it has a name, an accession number, the sequence of the protein and the source where this information has been taken.

The *Transcript* class and the *Protein* class are also represented in the conceptual schema by the *rna_e* class and the *protein_e* class respectively.

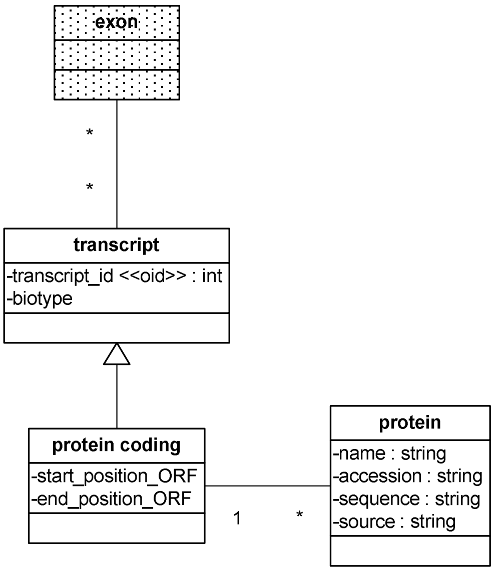


Fig. 2. Transcription view

3.3 Variation View

The Variation View (Figure 3) models the knowledge related to the differences in DNA sequence among individuals. It is widely accepted that there are not two persons genetically identical, even identical twins have some variations acquired during development. For this reason, the study of the genetic differences between individuals or populations becomes relevant for medical and evolutionary reasons.

The main class in this view is the *Variation* class, with an internal identifier (*variation_id*) and a description (*description*). Variations are specialized following two criteria: the precision in their description (*Description*), and its frequency (*Frequency*).

In hierarchy *Frequency*, a variation can be specialized in two types, variations with low frequency and pathologic effect (*Mutation*), and variations that appear in more than 1% of the population (*Polymorphism*).

Polymorphisms are specialized as CNV (copy number variation) or as SNP (single nucleotide polymorphism).

Copy number variations or CNVs are defined as variations consisting in abnormal repetitions or deletions of a nucleotide sequence. Recent discoveries reveal that CNV are the most prevalent and important form of genetic variation and they may have important roles in disease and drug response. The *CNV* class represents this concept, the multivalued attribute *repetitions* storing the usual repetitions.

On the other hand, a SNP is a polymorphism that occurs when a single nucleotide in the genome differs between individuals, many of them have no effect on gene function but others, as CNV, may predispose to disease or influence on the response to a drug. The *SNP* class represents this concept and the *map_weight* attribute is the number of times that this variation is found in the genome. Related to SNP variations, more information is modeled. The *SNP_Allele* class represents the different alleles of a SNP (attribute *allele* with domain {A, T, G, C}). The *SNP_Genotype* class represents the allele pairs of a SNP taking into account both homologue chromosomes (attributes *allele1* and *allele2*, with domain {A, T, G, C}).

Information about the frequency of SNP variations in different populations is also modeled, this kind of information is very important for population genetics studies, which have direct impacts in genetic counseling, forensic medicine and genetic screening. The *Population* class represents human groups with common features (*name*, *description* and *size* of each group are stored). The *SNP_Allele_Pop* and *SNP_Genotype_Pop* classes represent the frequency (*frecuency* attribute) of each SNP in each population.

Another concept modeled in the view is the *LD* class (linkage disequilibrium), that models the relation between two SNPs in one specific population with attributes *Dprime*, *Rsquare* and *LOD*. These are just indicators of the level of disequilibrium of that specific allele combination on a specific population. The amount depends of the difference between observed and theoretically random frequencies.

In the other hierarchy –complementary and orthogonal to the previous one-, *Descriptio*n, a variation is specialized into two classes, *Precise* and *Imprecise*.

Imprecise class represents variations whose unique known information is a description in natural language, which does not indicate position within the DNA sequence or base pair length.

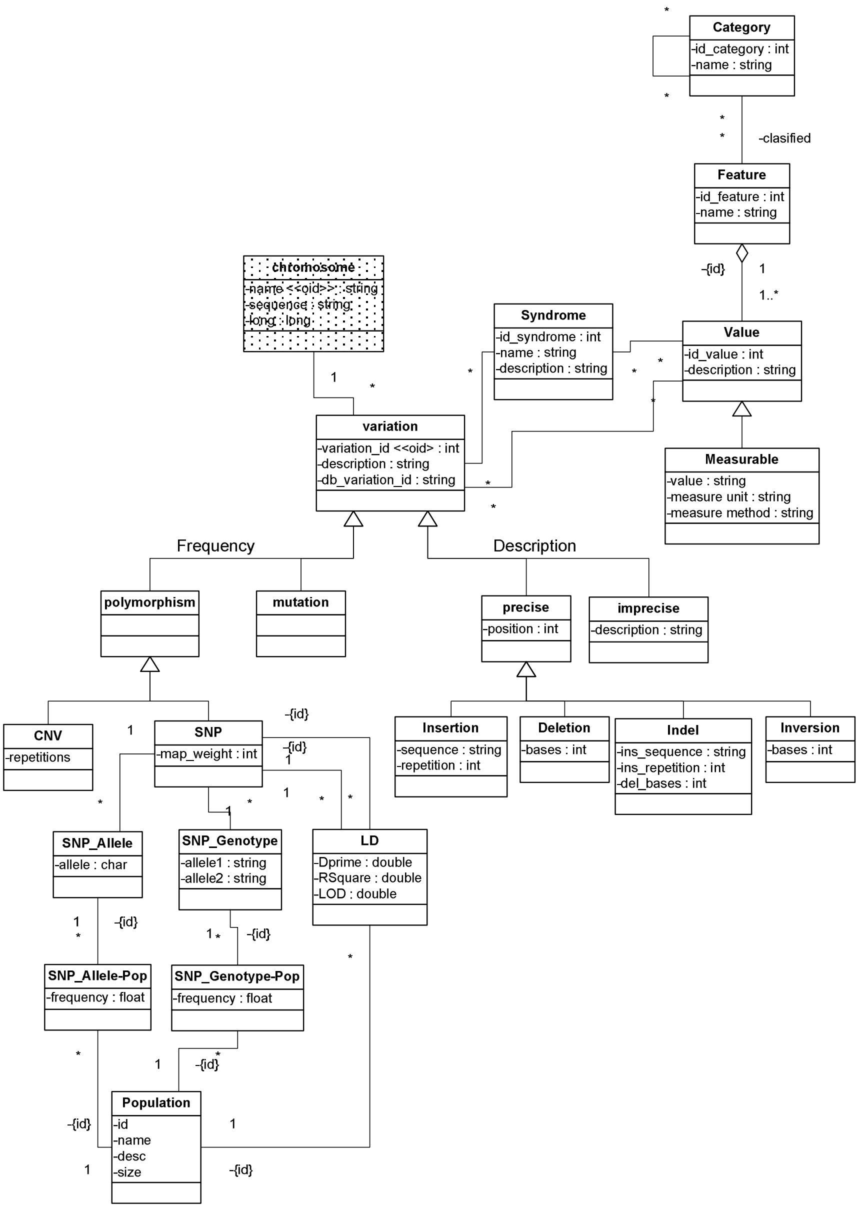


Fig. 3. Variation view

On the contrary, *Precise* class represents variations with known position (*position*) within the chromosome DNA sequence. *Precise* class is specialized into four new types: *Insertion*, *Deletion*, *Indel* and *Inversion*.

Insertion class represents variations consisting of insertions of nucleotide sequences (*sequence)* a number of times *(repetition*) in the chromosome DNA sequence; *Deletion* class represents variations consisting of deletions of a number (*bases*) of nucleotides; *Indel* class represents variations consisting of both insertions (*ins_sequence, ins_repetition*) and deletions (*del_bases*); and finally, *Inversion* class represents variations that causes a reversal in the order of a nucleotide sequence (*bases*) in the chromosome.

The *Category*, *Feature*, *Value*, *Measurable* and *Syndrome* classes associate a variation to a phenotype. *Syndrome* class corresponds to the general concept of disease; neurofibromatosis and Huntington´s are examples of instances of this class. A syndrome can be caused by one or several variations; and a variation can have multiple diseases associated to it. Usually, syndromes are characterized by various features, instances of the *Feature* class; in the case of neurofibromatosis, this includes the so-called *café au lait* spots features. These features in turn, are classified by categories, which have a recursive property indicated by the self-referencing relationship. Adding to this, each feature has an associated value, which is the measurable effect on phenotype (*Measurable* class). In the case of Huntington’s syndrome, this corresponds to the blood markers used to detect tumors. It is important to note that not every variation is associated to a specific phenotype; typically, variations characterized as polymorphisms do not cause pathological phenotypes.

3.4 Pathway View

Proteins, transcripts, chromosomal elements and many other molecules interact inside the cell in many different ways. This molecular interactions and reaction networks are generally called Metabolic Pathways. Through metabolic pathways the cell produces the energy and cell components needed for cell function. If metabolic pathways are altered, cell function might be compromised and this lead to disease. Examples of metabolic diseases are Phenilketonuria, Galactosemia or Tay-Sachs disease.

In the cell metabolism, some substances are broken down producing energy for vital processes while other substances necessary for life are synthesized. Pathways are important to the maintenance of homeostasis within an organism. Pathways are catalyzed by enzymes that regulate these reactions, and often require minerals, vitamins, and other cofactors in order to function properly. Since metabolic pathways usually involve many *metabolites*, they can be quite elaborated. Inside the cell an enormous number of distinct pathways exist; this collection of pathways is called the metabolic network.

A metabolic pathway is a sequential modification, or process, that transforms the initial metabolite into a final product, obtaining in the way energy or another kind of sub products. All these products resulting from the metabolism can be used immediately or be stored in the cell. There are also cases where the end product of a pathway is just the initial product of another one. This process composition is represented in the schema (Figure 4) by the *Event* class, and its specialized classes, *Process* and *Pathway*. *Process* class represents a single, atomic process and *Pathway* class represents a complex process formed by a sequence of complex and single processes. The association between *Event* and *Pathway* represents the pathway composition in its simpler event components; and the association with edges *Pre* and *Post* allows us to know the order of simpler events in a pathway composition.

An entity can take part in a process in three ways. In the first one, the entity is the principal chemical or one cofactor, that is, the necessary input for that process; sometimes these entities are also called the substrates. The second way, the entity is the result of that process, that is, the output or end-product. And the third and last way, the entity is a regulator of that process; we distinguish two kinds of regulation: activation and inhibition. There is a special kind of regulation, catalysis, which has been modeled apart due to the fact that in some processes the catalyst is unknown.

This behavior is modeled by the specialization of *Takes_part* class into *input*, *output* and *regulator* classes. The *stoichiometry* attribute of *input* and *output* classes represents the amount of entity that is involved in the process. The values for the *type* attribute of *regulator* class are *activator* and *inhibitor*, indicating the type of regulation. Due to the exception commented above, the catalysis *class* is included. In the situations that the catalyst is known, an enzime is associated with the corresponding process.

The entities that participate in a simple process from a metabolic pathway are specialized in several classes: *Simple, Polymer, Complex* and *EntitySet*. The *Simple* class represents the elementary entities; it is specialized in the *Gen_E*, *Rna_E, Protein_E, Aminoacid_E, Nucleotide_E* and *Basic_E* classes. The *Polymer* class represents entities that are generated by the repetition of some complex or simple entity; its *min* and *max* attributes represent the simple entity range of repetitions. *Complex* class models entities formed by the combination of some other simpler entities; its *detection_method* attribute indicates the technique used to detect complex formation. The class *Component* represents how a complex entity is formed by its simpler component entities; *stoichiometry* and *interaction* attributes allows us to know how and in which quantities the complex is formed by its components respectively. The *EntitySet* class represents the common way of participation of several entities in some process, and allows us to simplify the way we describe similar processes.

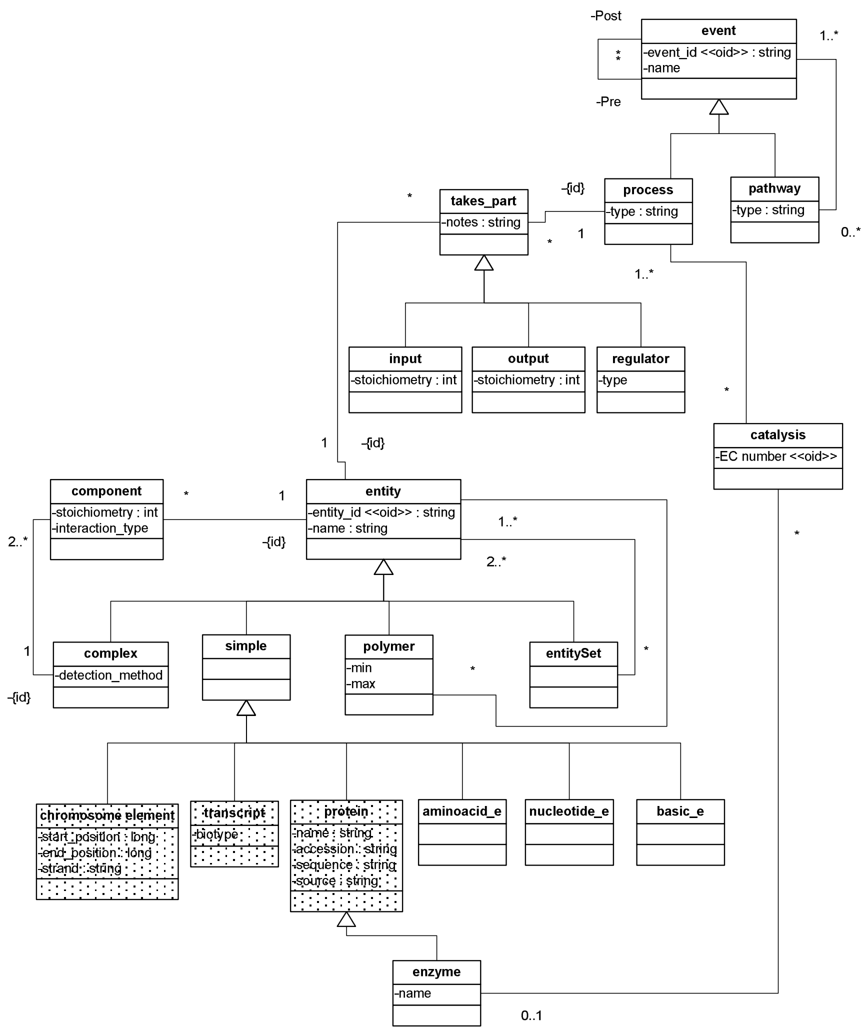


Fig. 4. Pathway view

3.5 Data Source and Bibliography Reference View

To maintain information about data sources and bibliographic references that might be interesting, the schema includes the following classes (Figure 5): *data_bank* (with *name* and *description* attributes*), data_bank_version* (with *release* and *date* attributes), *element_data_bank* (with *source_identification* which identifies the chromosome element in the data bank), *Bibliography DB* which represents the different sources where scientific publications can be found in the web, and *Bibliography reference* which represents the published articles. Relationships between this class and many classes in the schema (variation, exon, transcript, entity) can be found.

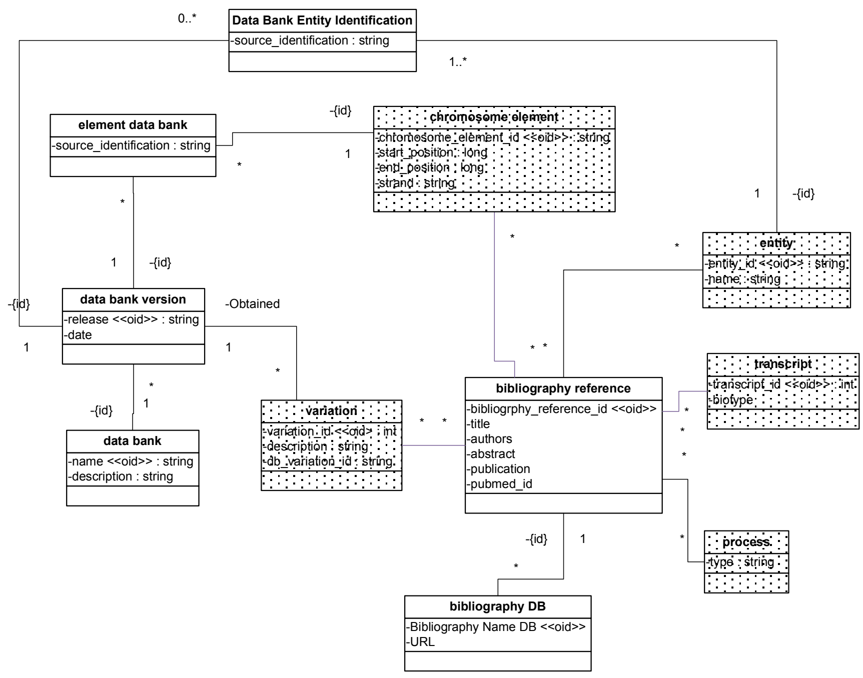


Fig. 5. Data source and bibliography reference view

4 Conclusions

In the recent times we are pursuing the goal of defining a Conceptual Schema of the Human Genome that could be used as a sound ontological background to provide some sort of “Genome Wikipedia” intended to capture all the valuable data that exists in the Bioinformatics domain. We introduced in this work the latest version of such an effort, where the main effort has been concentrated around integrating the different, relevant views that exist when dealing with genome information: the structural view, the transcription view, the variation view, the pathway view and the source & bibliography view.

All these views are normally treated separately, what generates heterogeneous and dispersed set of data sources, too often including inconsistencies, out-of-date information, redundancies, and that are at the end hard to manage and exploit efficiently. It is our position that only under the coverage of a complete Conceptual Schema that collects appropriately all the relevant concepts for all the five considered views, it will be possible to provide an answer to the adequate management of the huge amount of genomic data that is continuously being generated.

The Conceptual Schema presented in this chapter is a concrete answer to the quoted integration challenge, and it is our belief that it constitutes the only valid road to success: a proper link between the knowledge accumulated during decades in the Information Systems community, adapted to the idiosyncrasy of the human genome domain.

The way in which the knowledge of the human genome domain is captured and acquired by the IS experts is also one of the richest contributions of the work, together with the potential applications of the subsequent genome database obtained from the conceptual schema in the context of the genome-based personalized medicine, that will require a reliable data storage system. A very interesting description of the so-called "DNA revolution" and how it can affect human health systems can be found in [42], where what we need to know about our DNA and why it matters is explained in a brilliant way. It is well-known how faster and cheaper a complete sequence of human DNA will be obtained very soon [43]. But to manage this huge amount of information, new IS have to be designed and implemented. That type of systems will be only possible if genotype and phenotype are formally associated through a strong conceptual connection that only a precise, complete and correct Conceptual Schema can provide. Having the required database design as a logical consequence of that Conceptual Schema, it is now the time of finding where are the reliable data, and defining the corresponding data provenance-based mechanisms intended to store it accordingly for efficient exploitation.

Our current efforts concentrate on demonstrating in some concrete scenarios that the conceptual model-centric approach can be used effectively by biologists. The Conceptual Schema of the Human Genome has been transformed into its corresponding database, and a set of projects are being developed to load the data that correspond to a set of selected genes related with well-known clinical pathologies (i.e. NF1 for Neurofibromatosis,or BRCA1-2 for Breast Cancer), whose current genomic manipulation requires a manual, costly, and prone-to-error process. The selection of the relevant information (data provenance perspective) is the first problem to solve. Once the relevant stakeholders are identified, and the information that they use is determined, the design and implementation of the database load modules is the next step to overcome, the goal being having the data under the control of "our" Data Base of the Human Genome. Having the structural order guaranteed by the sound Conceptual Schema background, a software application is constructed, intended to generate the reports that compare DNA patient samples with the DNA information stored in the database. Once the data are captured and managed in the database, the benefits of having a Conceptual Schema-centered approach becomes evident, in terms of quality of data, support to schema evolution and efficiency of the report generation process. Finally, scalability is an issue: what we do now for a few genes, could be done for the whole genome, providing that the genomic information were available. This is a work that will take years, and that constitutes the direction of our present and future work.

References

[1] Olivé, A.: Conceptual Modelling of Information Systems. Springer, Heidelberg (2007)

[2] Falkenberg, E., Hesse, W., Lindgreen, W., Nilsson, E., Han, J., Rolland, C., Stamper, R., Van Assche, F., Verrijn-Stuart, A., Voss, K.: A Framework of Information System Concepts. IFIP (1998)

[3] Pastor, O., Molina, J.C.: Model-Driven Architecture in Practice. Springer, Heidelberg (2007)

[4] Thorisson, G.A., Muilu, J., Brookes, A.: Genotype-phenotype databases: challenges and solutions for the post-genomic era. Nature Reviews – Genetics 10 (2009)
[5] Stein, L.: Creating a bioinformatics nation. Nature 417, 119–120 (2002)
[6] Crowd sourcing human mutations (Editorial). Nature Genetics 43(4) (2011)
[7] The Gene Ontology Consortium.: Gene Ontology: tool for the unification of biology. Nat. Genet. 25, 25–29 (2000)
[8] `http://www.geneontology.org/` (accessed December 05, 2011)
[9] `http://www.ncbi.nlm.nih.gov/gene` (accessed December 05, 2011)
[10] Hubbard, T., et al.: The Ensembl genome database project. Nucleic Acids Research 30(1), 38–41 (2002)
[11] `http://www.ensembl.org` (accessed December 05, 2011)
[12] `http://www.ncbi.nlm.nih.gov/genbank/` (accessed December 05, 2011)
[13] `http://www.hgmd.org/` (accessed December 05, 2011)
[14] Cooper, D.N., Krawczak, M.: Human gene mutation database. Hum. Genet. 98(5), 629 (1996)
[15] Hamosh, A., Scott, A.F., Amberger, J., Valle, D., McKusick, V.A.: Online Mendelian Inheritance in Man (OMIM). Hum. Mutat. 15(1), 57–61 (2000)
[16] `http://www.ncbi.nlm.nih.gov/omim` (accessed December 05, 2011)
[17] `http://www.genome.jp/kegg/` (accessed December 05, 2011)
[18] Ogata, H., Goto, S., Sato, K., Fujibuchi, W., Bono, H., Kanehisa, M.: KEGG: Kyoto Encyclopedia of Genes and Genomes. Nucleic Acids Res. 27(1), 29–34 (1999)
[19] Apweiler, R., et al.: UniProt: the Universal Protein knowledgebase. Nucleic Acids Res. 32 (Database issue): D115-9 (2004)
[20] `http://www.uniprot.org/` (accessed December 05, 2011)
[21] Apweiler, R., et al.: The InterPro database, an integrated documentation resource for protein families, domains and functional sites. Nucleic Acids Res. 29(1), 37–40 (2001)
[22] `http://www.ebi.ac.uk/interpro/` (accessed December 05, 2011)
[23] Croft, D., et al.: Reactome: a database of reactions, pathways and biological processes. Nucleic Acids Res. (Database issue): D691-7 (2011)
[24] `http://www.reactome.org/ReactomeGWT/entrypoint.html` (accessed December 05, 2011)
[25] Pastor, O.: Conceptual Modeling Meets the Human Genome. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 1–11. Springer, Heidelberg (2008)
[26] Paton, W.N., Khan, S., Hayes, A., Moussouni, F., Brass, A., Eilbeck, K., Globe, C., Hubbard, S., Oliver, S.: Conceptual modeling of genomic information. Bioinformatics 16(6), 548–557 (2000)
[27] Ram, S., Wei, W.: Modeling the Semantics of 3D Protein Structures. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 696–708. Springer, Heidelberg (2004)
[28] Garwood, K., Garwood, C., Hedeler, C., Griffiths, T., Swainston, N., Oliver, S., Paton, W.: Model-driven user interface for bioinformatics data resources: regenerating the wheel as an alternative to reinventing it. Bioinformatics 7, 532, 1–14 (2006)
[29] Bornberg-Bauer, E., Paton, N.: Conceptual data modelling for bioinformatics. Briefings in Bioinformatics 3(2), 166–180 (2002)
[30] e-fungi Project, `http://www.cs.man.ac.uk/cornell/eFungi/index.html`
[31] Hedeler, C., Wong, H.M., Cornell, M.J., Alam, I., Soanes, D., Rattray, M., Hubbrad, S.J., Talbot, N.J., Oliver, S.G., Paton, N.: e-Fungi: a data resource for comparative analysis of fungal genomes. BMC Genomics 8, 426, 1–15 (2007)

[32] Pastor, O., Levin, A., Celma, M., Casamayor, J., Virrueta, A., Eraso, L.: Model-Based Engineering Applied to the Interpretation of the Human Genome. In: Kaschek, R., Delcambre, L. (eds.) The Evolution of Conceptual Modeling. LNCS, vol. 6520, pp. 306–330. Springer, Heidelberg (2011)

[33] Pastor, O., van der Kroon, M., Levin, A., Casamayor, J.C., Celma, M.: A Conceptual Modeling Approach to Improve Human Genome Understanding. In: Embley, D., Thalheim, B. (eds.) Handbook of Conceptual Modeling: Theory, Practice and Research Challenges, pp. 517–541. Springer, Heidelberg (2011)

[34] International Human Genome Sequencing Consortium: Initial sequencing and analysis of the human genome. Nature 409(6822), 860–921 (2001)

[35] Gene Nomenclature Committee, `http://www.genenames.org`

[36] National Center for Biotechnology Information, `http://www.ncbi.nlm.nih.gov`

[37] Gerstein, M.B., Bruce, C., Rozowsky, J., Zheng, D., Du, J., Korbel, J., Emanuelsson, O., Zhang, Z., Weissman, S., Snyder, M.: What is a gene, post-ENCODE? History and updated definition. Genome Res. 17, 669–681 (2007)

[38] Blitzblau, H.G., Bell, G.W., Rodriguez, J., Bell, S.P., Hochwagen, A.: Mapping of Meiotic Single-Stranded DNA Reveals Double-Strand-Break Hotspots near Centromeres and Telomeres. Current Biology 17(23), 2003–2012 (2007)

[39] Paigen, K., Petkov, P.: Mammalian recombination hot spots: properties, control and evolution. Nature Reviews Genetics 11, 221–233 (2010)

[40] Holmquist, G.P.: Chromosome bands, their chromatin flavors, and their functional features. Am. J. Hum. Genet. 51(1), 17–37 (1992)

[41] Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P.: Molecular Biology of the Cell. Garland Science, New York (2002), `http://www.ncbi.nlm.nih.gov/books/bv.fcgi?rid=mboc4`

[42] Collins, F.S.: The Language of Life: DNA and the Revolution in Personalized Medicine. Harper Colllins Publishers (2010)

[43] Wheeler, D.A., et al.: The complete genome of an individual by massively parallel DNA sequencing. Nature 452, 872–877 (2008)

Transforming Geometrically Enhanced Conceptual Model Schemas to GML

Hui Ma

Victoria University of Wellington, New Zealand
hui.ma@ecs.vuw.ac.nz

Abstract. Successful implementation of geographic applications starts with conceptual design. A conceptual schema will then be transformed into a database schema that can be implemented. Geography Markup Language (GML) has emerged as an open standard that provides a common grammar for coding geo-spatial content and exchanging over the Internet. In this paper we discuss the transformation from Geometrically enhanced ER model (GERM) to GML. GERM is an extension of the classical ER model that has been successfully used for conceptual modelling of geographic applications. The transformation rules have been chosen such that relevant application semantics is preserved during the transformation. We further present an bottom-up algorithm for transforming GERM schemas into their GML counterparts. A case study is conducted to demonstrate the effectiveness of the algorithm.

1 Introduction

The Geography Markup Language (GML) is an XML-based language defined by the Open Geospatial Consortium (OGC) for storing and transporting geographic information. With the increasing number of web-based geographic information systems, GML is becoming the industry standard for exchanging and sharing information between geographic applications distributed across the Internet. For complex geographic applications, however, the creation of an adequate GML database schema is not easy. It is meanwhile regarded as best practise to design geographic information first at the conceptual level, and then to transform the conceptual schema into GML. Various conceptual modelling languages have been proposed for designing geographic information in the literature [7,8,13,24,25,26]. However, none of them comes without deficiencies, for a detailed discussion we refer to [19].

Motivated by the author's work on the sustainable land use initiative (SLUI) of the New Zealand government, we have recently introduced a geometrically enhanced ER model (GERM) as our approach to the conceptual modelling of geographic information [20]. The SLUI initiative which addresses environmental problems in New Zealand's agricultural and silvicultural regions. Whole farm plans (WFP) are a common tool to integrate environmental goals with current farming operations [1,21]. Based on an assessment of available natural resources, environmental issues are identified and evaluated, and countermeasures are developed. This task involves the capture and analysis of data from distributed data collections such as image data, classification data, spatial data, observational data, climate data, soil data, air pollution data, ecology data, vegetation distribution data, biodiversity data, and business data. The need to adequately model and process geometric properties of data objects involved (i.e., features such as farms, paddocks, buildings, trails, water resources) led us to propose an extension of the popular ER model that supports database designers in dealing with the geometry of objects to be represented in the database application.

We found GERM useful for various reasons. Firstly, GERM preserves the aggregation-based approach [12] of the ER model by means of (higher-order) relationship types [27], thus naturally supporting hierarchical structure. Secondly, GERM allow roles in relationship types to use bulk and choice constructors, thus supporting entity sets, lists, multisets, options and alternatives to occur as components of relationships. Using GERM, geometric properties in the application domain can be modelled by attributes that have geometric data types assigned to them. This defines the syntactic layer of GERM that largely remains within the popular ER framework, thus enabling a smooth integration with non-geometric conceptual models. It allows data architects to cope with modelling tasks that involve geometry in a familiar, non-challenging way thereby preserving all the positive experience made with ER modelling. The syntactic layer of GERM is complemented by an internal layer where geometric properties are represented as point sets. Thus, common geometric shapes like lines, rectangles, polygons, circles, Bézier curves, or Bézier patches can be captured in a most natural way. On its internal layer, GERM makes use of an extended algebra that modifies the standard Boolean operators (i.e., union, intersection, difference, complement) on point sets to achieve a higher degree of accuracy for derived geometric properties [19].

Once a conceptual schema has been created (e.g., using GERM) that captures the data needs of some geographic application under development, it has to be transformed into a data model on implementation level that can be stored and manipulated by a DBMS. Today, most popular DBMS have spatial extensions that use GML to import/export data. In this paper we will discuss the transformation from GERM to GML. The mapping from a GERM schema to GML will be guided by a set of transformation rules that ensure that the resulting GML schema conforms to the OGS standard [22].

Our paper is organised as follows. Section 2 reviews relevant contributions on the transformation from conceptual models to GML found in the literature. In Section 3 we briefly summarize basic ideas of GERM and GML that are essential for the transformation. In Section 4 we introduce rules for the transformation from GERM to GML. We use examples from whole farm plan modelling to illustrate our approach. Finally, Section 5 concludes the paper and makes suggestions for future research.

2 Related Work

In the literature, various approaches have been proposed for transforming conceptual schemas to XML. Most of them assume the input schema to be given in some variant of the classical ER model (such as EER or UML class diagrams). The target schema is then generated using the W3C standards XML DTD or XML Schema. For example, [14] proposes an algorithm for transforming EER schemas to DTDs, while [15] discusses the transformation of UML class diagrams to DTDs. [5,18,23] study the transformation of ER/EER schemas to XSDs. [2] introduces a new conceptual modelling language for hierarchical data (called C-XML) and studies the transformation of C-XML schemas to XSDs. All these proposals differ in the amount of application semantics that they preserve during the transformation. [17] discusses which parts of the application semantics captured in an ER schema can be preserved by a DTD or XSD.

With the increasing popularity of GML as a standard for representing and exchanging geographic information, researchers have started to investigate how to tailor the general transformation approaches above to the special case of geographic applications. Here the focus is on how to best match spatial extensions of conceptual modelling languages with constructs defined by GML. [4], for example, discusses the mapping of UML class diagrams with additional stereotypes for geographic information to GML 2.0. The proposed transformation rules generate flat GML schemas as all relationships between classes are transformed into GML elements located directly under the root element, thus neglecting the opportunities offered by XML for capturing hierarchical structures and resulting in poor query performance. Furthermore, cardinality constraints are ignored completely.

More recently, [11] presents a method for mapping conceptual schemas for geographic applications defined as extended UML class diagrams (called OMT-G) to GML 3.0. Here, relationships are transformed by nesting of GML elements, thus ensuring a better query performance. A case study is conducted to demonstrate that the nested approach performs better than the flat approach proposed in [4]. Based on [5], a table is presented showing how the classes participating in a relationship be transformed into nested GML elements depending on their associated cardinality constraints. However, in practice there are various ways of nesting and the paper gives no indication on which one to choose.

In this paper we will extend the discussion to GERM and the additional constructs it offers for data architects such as higher-level relationships and relationships with complex components (e.g. set-valued). Our transformation rules for mapping GERM schemas to GML make use of the capabilities of XML to capture hierarchical structure. In addition, we choose nestings that are likely to optimize query performance during the runtime of the geographic application, thus providing better assistance for data architects.

3 GERM and GML

Before we discuss the transformation from GERM to GML, we will briefly summarise basic ingredients of both languages that will be used later on.

3.1 GERM

GERM is an extension of Bernhard Thalheim's higher-order ER model [27] with emphasis on integrated geometric modelling. It preserves the aggregation-based modelling principle of the ER model, where a schema $\mathcal{S}$ consists of a set of object types. Object types are characterised by their attributes. We start with a countable set $\mathcal{U}$ of simple attributes (called the *universe*) together with a type assignment $tp(A)$ that assigns to each attribute $A \in \mathcal{U}$ its data type $tp(A)$. Complex attributes may be built from simple ones by nesting. Let $\mathcal{A}$ be the smallest superset of $\mathcal{U}$ such that

$$X(A_1, \dots, A_n), X\{A\}, X[A], X\langle A\rangle, X_1(A_1) \oplus \cdots \oplus X_n(A_n), X(A_1 \to A_2) \in \mathcal{A}$$

holds whenever $A, A_1, \dots, A_n \in \mathcal{A}$ holds, with labels $X, X_1, \dots, X_n$ chosen from some fixed alphabet $\mathcal{L}$. The type assignment tp extends naturally from $\mathcal{U}$ to $\mathcal{A}$ as follows:

- $tp(X(A_1, \dots, A_n) = (a_1 : tp(A_1), \dots, a_n : tp(A_n))$ with labels $a_1, \dots, a_n \in \mathcal{L}$,
- $tp(X\{A\}) = \{tp(A)\}$, $tp(X[A]) = [tp(A)]$, $tp(X\langle A\rangle) = \langle tp(A)\rangle$,
- $tp(X_1(A_1) \oplus \cdots \oplus X_n(A_n)) = (X_1 : tp(A_1)) \oplus \cdots \oplus (X_n : tp(A_n))$, and
- $tp(X(A_1 \to A_2)) = tp(A_1) \to tp(A_2)$.

The key extension of GERM, however, is the presence of geometric domains for geometric data types in $\mathcal{A}$. The data type $tp(A)$ assigned to a (base) attribute $A \in \mathcal{U}$ is some *base type*, like INT, $FLOAT$, $STRING$, $DATE$, or $TIME$. Each base type t is associated with a countable set of values $dom(t)$ called the *domain* of t. For the base types mentioned here the domains are chosen as usual. For an attribute $A \in \mathcal{U}$ we let $dom(A) = dom(tp(A))$, and also call $dom(A)$ the *domain* of A. In GERM, we can use constructors to build *complex data types* t from base types. In particular, we use $(\cdot)$ for record types, $\{\cdot\}$, $[\cdot]$ and $\langle\cdot\rangle$ for finite set, list and multiset types, respectively, $\oplus$ for (disjoint) union types, and $\to$ for map types. We may also use a trivial type $\mathbb{1}$ with domain $dom(\mathbb{1}) = \{\perp\}$. Generally, we allow complex data types to be named and used in type definitions in the same way as base types with the restriction that cycles are forbidden. Domains associated with complex data types are then obtained in a similar way from the values of the respective base types and $\perp$.

Note that complex data types offer additional opportunities for data architects but are not a must-use. Consider for example a geometric property *shape* whose values are polygons. Polygons may be seen as lists of points. Points again may be seen as pairs of real numbers (assuming a two-dimensional Euclidean plane). If the data architect considers the internal structure of polygons to be conceptually irrelevant for the application under development then she can define a simple attribute *shape* with $tp(shape) = POLYGON$. Otherwise, if the internal structure of polygons is conceptually relevant then she may use a complex attribute $shape([point])$ with $tp(point) = POINT$ instead. If in addition the internal structure of points is considered relevant, too, then she may even use a complex attribute $shape([point(x\text{-}coord, y\text{-}coord)])$ with $tp(x\text{-}coord) = tp(y\text{-}coord) = FLOAT$.

This indicates that points and their coordinates are conceptually relevant beyond representing a data type.

Example 1. In GERM, data architects may define named complex data types to be used for modelling geometric properties. Examples include dedicated data types such as $\mathit{Point} = (x : \mathit{FLOAT}, y : \mathit{FLOAT})$ for points in the two-dimensional plane, $\mathit{PolyLine} = [\mathit{Point}]$, $\mathit{Polygon} = [\mathit{Point}]$, $\mathit{Bezier} = [\mathit{Point}]$, or $\mathit{PolyBezier} = [\mathit{Bezier}]$. Note, that the examples *PolyLine*, *Polygon*, and *Bezier* constitute types with identical surface representations. A *polyline* is defined defined piecewise linearly, while a *polygon* is a region with a polyline border. A *Bézier curve* is determined by a finite sequence of points in the plane, too. It passes through the first and last control points and lies within the convex hull of the control points. A *polyBézier curve* is defined piecewise by Bézier curves. We will commonly refer to such data types as *geometric data types*, thus emphasising that they will serve as surface representations for particular geometric properties.

The trivial type $\mathbb{1}$ can be used in combination with the union constructor to define *enumerated types*, i.e., types with finite domains, such as $\mathit{Bool} = (\mathbf{T} : \mathbb{1}) \oplus (\mathbf{F} : \mathbb{1})$, $\mathit{Gender} = (\text{male} : \mathbb{1}) \oplus (\text{female} : \mathbb{1})$, or $\mathit{INT}_n = (1 : \mathbb{1}) \oplus \cdots \oplus (n : \mathbb{1})$ for any positive integer n, which gives a domain with values $1, \dots, n$.

The map constructor can be used to define *array types*, such as $\mathit{Patch} = (i : \mathit{INT}_n, j : \mathit{INT}_m) \to \mathit{Point}$ representing Bézier patches. Further examples used for spatial modelling are *vector field types* of different dimensions, such as $\mathit{Vectorfield1} = \{\mathit{Point}\} \to \mathit{FLOAT}$, which is useful for capturing sensor data (e.g., water levels), and $\mathit{Vectorfield2} = \{\mathit{Point}\} \to \mathit{Point}$, which is useful for modelling other measurements (e.g., wind force and direction) by two-dimensional vectors. Finally, $\mathit{TimeSeries} = (d : \mathit{DATE}, t : \mathit{TIME}) \to \mathit{Vectorfield1}$ is useful for modelling a series of observed data over time, thus capturing also temporal aspects of data. $\square$

Similar to Thalheim's higher-order ER model, GERM allows the nesting of relationships (that is, relationships may participate in other relationships) and cluster components. We extend this idea further as follows. Let $\mathcal{O}$ be a countable set of object type names. The set $\mathcal{C}$ of *component expressions* (over $\mathcal{O}$) is the smallest set containing all object type names $O \in \mathcal{O}$, all list expressions $[C]$, all set expressions $\{C_0\}$, all multiset expressions $\langle C_0 \rangle$, and all union expressions $C_1 \oplus \cdots \oplus C_n$, whenever $C, C_0, C_1, \dots, C_n \in \mathcal{C}$ holds, but such that the C_i are not union expressions.

A *structured component* is a pair $\rho : C$ with a *role name* ρ and a component expression $C \in \mathcal{C}$. Let l be a natural number. An *object type* O of level l consists of a finite set $\mathit{comp}(O) = \{\rho_1 : C_1, \dots, \rho_n : C_n\}$ of structured components with pairwise different role names $\rho_1, \dots, \rho_n$, and a finite set $\mathit{attr}(O) = \{A_1, \dots, A_m\} \subseteq \mathcal{A}$ of complex attributes. Each object type that occurs in any of the component expressions C_i is of level at most $l - 1$, and at least one of these object types must have exactly the level $l - 1$ (unless $\mathit{comp}(O) = \emptyset$).

By this definition, O is of level 0 if and only if $\mathit{comp}(O) = \emptyset$ holds. Therefore, object types of level 0 are called *entity types*, while object types of level $l > 0$ are called *relationship types*. For brevity, we use the notation $O = (comp(O), attr(O), key(O))$ to define an object type O. For simplicity, we assume that the primary key of O consists of attributes and structured components of O, that is, $key(O) \subseteq comp(O) \cup attr(O)$.

It is common a practice to visualise entity-relationship schemas as diagrams. This approach has also been used in the presence of higher-level relationships (that is, object types of level $l \geq 2$), see [27]. We will adopt this approach to our purposes here. The *GERM diagram* of a GERM schema S is a directed graph with the object types of S as nodes, and with edges from a node O to a node Q whenever Q appears in a structured component $\rho_i : C_i \in comp(O)$. As suggested in [27] disjoint unions (also called clusters) are indicated by attaching a $\oplus$-symbol to the diamond on the edge to the relationship type. Moreover, for our examples here we indicate sets by attaching a $\otimes$-symbol to the diamond on the edge to the relationship type, see for example Figure 1.

Next, we adapt participation cardinality constraints defined in [27] to GERM. Note that participation cardinality constrains are only defined on relationship types, which are object types at level 1 and above.

Definition 1 (Participation cardinality constraint). Let O be a relationship type in a GERM schema $\mathcal{S}$, and let X_i be a component of O, say in the form O_i or $p_i : O_i$.

1. A *participation cardinality constraint* on a relationship type O is an expression of the form $card(O, X_i) = (a, b)$ where a is a natural number, and b is a natural number or ∞.
2. An instance $\mathcal{I}$ of the GERM schema $\mathcal{S}$ is said to *satisfy* the participation cardinality constraint $card(O, X_i) = (a, b)$ if and only if for all objects $o_i \in \mathcal{I}(O_i)$ we have
$$a \leq \#\{o \in \mathcal{I}(O) \mid o(X_i) = o_i\} \leq b,$$
that is, each object o_i of type O_i participates in at least a and in at most b relationships of type O. □

We write $\mathcal{I} \models card(O, X_i) = (a, b)$ to denote that an instance I satisfies a given cardinality constraint $card(O, X_i) = (a, b)$.

Example 2. Let us consider a GERM schema for modelling *whole farm plans* (WFP) as illustrated by its diagram in Figure 1. Among other it reflects geographic information related with the farms that are managed with whole farm plans. The corresponding GERM database schema consists of the following object types:

– FARM = ({ *farm_name: STRING*, *owner: STRING*, *contact: STRING*, *boundary: PolyBezier*}, { *farm_name*})

– PADDOCK = ({ *in:* FARM }, { *p_code: STRING*, *boundary: PolyBezier*, *usage:* $(cattle : \mathbb{1}) \oplus (dairy : \mathbb{1}) \oplus (hort : \mathbb{1}) \oplus (sheep : \mathbb{1}) \oplus \cdots \oplus (other : \mathbb{1})$}, { *in, p_code* })

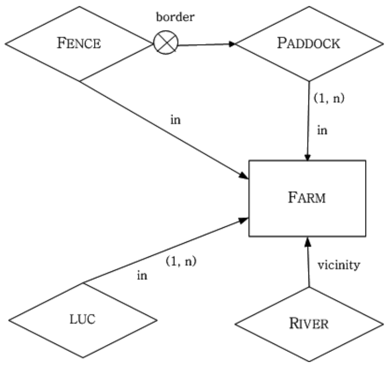


Fig. 1. Example of a GERM diagram for whole farm plan modelling

- FENCE = ({ *in:* FARM, *border:* {PADDOCK}}, { *shape: PolyLine*}, { *in, shape* })
- RIVER = ({ *vicinity:* FARM }, { *river_name: STRING*, *left: PolyBezier*, *right: PolyBezier*}, { *vicinity, river_name* })
- LUC = ({ *in:* FARM }, { *luc_no: INT*, *year: YEAR*, *boundary: PolyBezier*, *unit(class: INT*, *sub_class:* $(w : \mathbb{1}) \oplus (e : \mathbb{1}) \oplus (s : \mathbb{1}) \oplus (c : \mathbb{1})$, *capability: INT*)}, { *luc_no* })

Whole farm plans are developed on the basis of paddock maps that exits for every farm. Each paddock belongs to a farm, is identified within the farm by its *p_code*, is legally defined by a *boundary*, and has a particular *usage*. We can use an object type PADDOCK with structured component *in* : FARM and with attributes *p_code*, *boundary*, and *usage* to model paddocks. That is, *comp*(PADDOCK) = {in : FARM} and *attr*(PADDOCK) = {*p_code, boundary, usage*}. Herein, FARM is an entity type and, thus, PADDOCK is a relationship type of level 1.

Furthermore, the following cardinality constraints have been defined on the schema:

- *card*(PADDOCK, *in* : FARM) = $(1, n)$,
- *card*(LUC, *in* : FARM) = $(1, n)$. □

3.2 GML

GML is an XML-based standard developed by the OpenGIS consortium (OGC) that is used for representing and exchanging geographic information [22] on the internet, including non-spatial and spatial properties of geographic features. GML enables us to store non-spatial and spatial data in the same database, see e.g. Oracle Spatial, DB2 Spatial, or PostgreSQL [16].

Similar to GERM, GML separates concepts from data types. Using GML, the schema for an application's geographic data is created as an XSD (called the *application schema* [22]) that is specific for the application's domain of interest. GML distinguishes *features* from *geometry objects*, cf. [22]: A *feature*, which may or may not have geometric aspects, is an application object that represents a physical entity (such as a farm, a river, or a person), while a *geometry object* defines a location or region instead of a physical entity, and hence is different from a feature.

Note that there are still many geographic information systems (GIS) in use that do not explicitly distinguish between features and geometry objects (as done in GML and also in GERM), but regard them interchangeably as items on a map. GML overcomes this issue, thus providing a clear guideline for data architects. The GML standard itself defines a collection of geometry types that may be used by data architects, and whose meaning is independent of the particular applications (thus enabling data sharing and integration). The GML schema for a particular application (that is, the application schema) imports the general GML standard and thus can use the defined geometry types. In the GML schema, the feature types are defined, and these definitions may use the available geometry types to represent geometric properties. This allows data architects to cope with modelling tasks involving geometric properties in the same way as with modelling tasks where no geometry comes into play. This is consistent with the foundations of GERM. Recall that in a GERM schema, object types with geometric properties can be equipped with attributes that have geometric data types assigned to them. Note that the explicit distinction between concepts and (geometry) data types in GERM, enables a much simpler transformation of GERM schemas to GML schemas, than for other conceptual models (e.g., OMT-G [11]).

GML supports various geometry types in addition to base data types in XML, including *Point*, *LineString*, *LinearRing*, *Box*, *Polygon*, *MultiPoint*, *MultiLineString*, and *MultiPolyn*. In addition to these types (that were already available in GML 2.0), the current version of the GML standard (GML 3.0) supports many new geometry types such *Arc*, *Circle*, *CubicSpline*, *Ring*, *OrientableCurve*, *MultiSurface*, *MultiCurve*, *MultiSurface*, *CompositeCurve*, *CompositeSurface*, or *CompositeSolid*.

Data types assigned to attributes reflecting geometric properties in GERM can be transformed directly to the corresponding data types in GML. For example, we can transform *PolyLine* in GERM to *LineString* in GML, and *Bezier* to *BezierType* in GML.

4 Generating a GML Schema from a GERM Schema

In this section, we will first present mapping criteria that should be considered while transforming a GERM schema to a GML schema. We will then present a set of transformation rules that comply with the mapping criteria, followed with a transformation algorithm.

4.1 Transformation Criteria

Several transformation criteria have been discussed in the literature mainly for transforming a conceptual schema to a target schema in XML. The most important is *semantic preservation* which requires that the transformation should preserve the semantics contained in the input schema, so that no semantic information contained in the conceptual schema is lost [11,18]. Secondly, *absence of redundancy* is also required, that is, there should be no data redundancy in XML documents that conform to the target schema, so that data consistency can be ensured [18]. It is further recommended to *use highly nested structure*, so that navigation along paths can be processed efficiently. Furthermore, *reversibility of design* [18] is required to ensure that the input schema can be recomputed from the target XML schema. There has been only very limited research concerning the transformation between conceptual schemas for geographic applications and XML/GML, cf. [6,11]. In this paper, we will adapt the transformation criteria mentioned above when discussing the transformation from GERM to GML.

4.2 Transformation Rules

Now, we will present a set of transformation rules that are inspired by the transformation criteria mentioned above. Throughout, let $\mathcal{S}$ be a GERM schema. The GML schema to be constructed from $\mathcal{S}$ will be denoted by $\mathcal{T}$. The transformation rules will serve to map the input schema $\mathcal{S}$ to the target schema $\mathcal{T}$. To begin with, let $\mathcal{T}$ be empty.

Rule 1 (Create root element). *Create a new XML element R with an appropriate element name and insert it into $\mathcal{T}$ as the root element.*

```
<xs:element name="WFP">
        <xs:complexType>
                <xs:complexContent>
                        <xs:extension base="gml:AbstractFeatureCollectionType">
                                <xs:sequence>
                                        <xs:element name="Farm">
                                                ...
                                                <xs:attribute name="id" type="xs:int"/>
                                                ...
                                        </xs:element>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
</xs:element>
```

Fig. 2. Example of creating root element and transforming entity types

In the sequel, let *id* be a valid attribute name in XML that does not occur in the conceptual schema.

Rule 2 (Transform entity types). *For an object type O of level 0 in $\mathcal{S}$, create a new XML element O and insert it into $\mathcal{T}$ as a child of the root element R. Then, equip O with a new simple attribute id.*

For example, to transform the WFP GERM Schema in Example 2 we use Rule 1 and Rule 2 to insert a root element WFP and to translate entity type FARM into a child FARM of the root element WFP in the output GML schema, cf. Figure 2.

For relationship types, on the other hand, the general idea is to place them as a child of one of its components. When a relationship type has more than one component, we need to decide which component will serve as its parent element in the target schema. The resulting GML schema should be efficiently accessible by as many queries as possible. Therefore it is reasonable to choose the component that is more often accessed together with the relationship type under consideration. We propose to use *Object Type Affinity* to measure how often the relationship type is accessed together with any of its components.

Definition 2 (*Object Type Affinity*). Consider a set of queries $Q^m = \{q_k : k = 1, \ldots, m\}$ with frequencies f_k $(k = 1, \ldots, m)$. For an object type O_i, let $use(q_k, O_i) = 1$ if it is accessed by query q_k, and 0 otherwise. For two object types O_i, O_j, we define their *affinity* $aff(O_i, O_j)$ as the sum of frequencies of all queries q_k $(k = 1, \ldots, m)$ accessing both O_i and O_j simultaneously:

$$aff(O_i, O_j) = \sum_{k=1}^{m} f_k \cdot use(q_k, O_i) \cdot use(q_k, O_j)$$

Definition 3 (*Owner Component*). Let O be a relationship type with structured components O_i or $\rho_i : O_i$ $(i = 1, \ldots, n)$. The *owner component* C^O_{owner} is the object type O_i $(i = 1, \ldots, n)$ that has maximal affinity with O, i.e.,

$$aff(O, C^O_{owner}) = \max_{i=1..n} aff(O, O_i).$$

If a relationship type has one structured component, i.e. $i = 1$ then the structured component is the owner component of the relationship type O, i.e.,

$$C^O_{owner} = O_1.$$

If there are two or more O_i that have the same maximal affinity with O, then the owner component is chosen among them.

Rule 3 (Transform relationship types). *For an object type O of level $l \geq 1$ in $\mathcal{S}$, create a new XML element O and insert it into $\mathcal{T}$ as a child of the XML element corresponding to its owner component C^O_{owner}. For every component C of O other than the owner component, create a new XML element C' and insert it into $\mathcal{T}$ as a child of O. Then, equip O as well as each of its children C' with a new attribute id.*

For example, consider the relationship type PADDOCK. It has only a single component FARM, which is therefore its owner component. Hence, the relationship type PADDOCK will be transformed into a child of the XML element FARM in the GML schema, cf. Figure 3.

```
<xs:element name="Farm">
        <xs:complexType>
        <xs:complexContent>
        <xs:extension base="gml:AbstractFeatureType">
        <xs:sequence>
                <xs:element name="farm_name" type="xs:string"/>
                <xs:element name="owner" type="xs:string"/>
                <xs:element name="contact" type="xs:string"/>
                <xs:element name="boundary" type="gml:BezierType"/>

                <xs:element name="Paddock" maxOccurs="unbounded">
                        <xs:complexType>
                        <xs:sequence>
                                <xs:element name="p_code" type="xs:string"/>
                                <xs:element name="boundary" type="gml:BezierType"/>
                                <xs:element name="usage" type="xs:string"/>
                        </xs:sequence>
                        <xs:attribute name="id" type="xs:int"/>
                        </xs:complexType>
                </xs:element>
                …
        </xs:sequence>
        <xs:attribute name="id" type="xs:int"/>
        </xs:extension>
        </xs:complexContent>
        </xs:complexType>
        …
</xs:element>
```

Fig. 3. Example of transforming relationship types

Rule 4 (Transform participation cardinality constraints). *For an object type O of level $l \geq 1$ in $\mathcal{S}$, if there is a participation cardinality constraint $card(O, C^O_{owner}) = (a, b)$ defined on $\mathcal{S}$ then set $minOccurs$ = "a" and $maxOccurs$ = "b" for the child O of C^O_{owner} in $\mathcal{T}$.*

In the example above the participation cardinality constraint $card$(PADDOCK, in : FARM) $= (1, n)$ is transformed into $minOccurs$ = "1" (which can be omitted in the GML schema as it is the default value) and $maxOccurs$ = "*unbounded*" for the child PADDOCK of the element FARM (as seen in Figure 3).

Rule 5 (Transform complex attributes). *For a complex attribute A of an object type O in $\mathcal{S}$, create a new XML element A and insert it into $\mathcal{T}$ as a child of the XML element O.*

The data type assigned to the complex attribute in GERM is transformed into the corresponding data type in GML.

For example, the data types *Point*, *Polyline*, *Polygon*, *PolyBezier*, and *Circle* discussed above will be transformed to the complex geometry types *PointType*, *LineStringType*, *PolygonType*, *BezierType*, and *CircleType* available by GML.

Next, we discuss the transformation of primary keys of the object types in the conceptual schema. Consider an entity type O. The straightforward approach is to define a key declaration on the root element R of $\mathcal{T}$ such that its key selector picks the child O of R and such that its key fields pick the grandchildren of R corresponding to the key attributes of O. Unfortunately, we face a limitation of the XML Schema standard (and thus of GML, too), namely that key fields may only pick items (XML elements or attributes) that are of simple data type. The same limitation needs to be respected for object types of higher level. In our example above, the object type FENCE has a key attribute *Shape* that is not simple. We call the primary key of an object type *transformable* if it contains no attribute that is not simple. Transformable primary keys are easy to preserve during the transformation from GERM to GML. If a primary key is not transformable then one might consider some work-around, e.g., replace non-simple key attributes by a suitable set of simple ones, or use a surrogate key instead. Also, note that in the literature on XML more flexible definitions of key constraints have been discussed that allow also non-simple data types for the key fields of an XML key [3]. For a recent discussion we refer to [9,10].

Rule 6 (Transform key constraints). *For an object type O in $\mathcal{S}$, insert a key declaration defined on the root element R in $\mathcal{T}$ such that its key selector picks the XML element corresponding to O in $\mathcal{T}$ and such that its single key field picks the attribute id of O.*

Furthermore, for an object type O of level $l \geq 1$ in $\mathcal{S}$, if its primary key is transformable and does not contain the owner component of O, insert a key declaration defined on the root element R in $\mathcal{T}$ such that its key selector picks the XML element corresponding to O in $\mathcal{T}$ and such that its key fields pick the key attributes of O and for each key component C the attribute id of the corresponding child C' of O in $\mathcal{T}$.

Furthermore, for an object type O of level $l \geq 1$ in $\mathcal{S}$, if its primary key is transformable and contains the owner component C^O_{owner} of O, insert a key declaration defined on the XML element corresponding to C^O_{owner} in $\mathcal{T}$ such that its key selector picks the XML element corresponding to O in $\mathcal{T}$ and such that its key fields pick the key attributes of O and for each key component C of O (other than C^O_{owner}) the attribute id of the corresponding child C' of O in $\mathcal{T}$.

Finally, we aim to preserve referential integrity when transforming the GERM schema $\mathcal{S}$ to GML. Due to the tree structure of GML, we need to make foreign key constraints explicit in the target schema $\mathcal{T}$.

Rule 7 (Assure referential integrity). *For an object type O of level $l \geq 1$ in $\mathcal{S}$, insert a keyref declaration defined on the XML element O in $\mathcal{T}$ for each component C of O (other than the owner component C^O_{owner}) such that its key selector picks the child C' of O in $\mathcal{T}$, its single key field picks the attribute id of C', and such that it references the respective key declaration inserted for the XML element C in $\mathcal{T}$.*

4.3 Mapping Procedure

Based on the transformation rules above, the entire mapping of the GERM schema $\mathcal{S}$ (input schema) to a GML schema $\mathcal{T}$ (target schema) will be conducted with the following algorithm.

Algorithm 1 (GERM-to-GML Mapping Algorithm)

```
Input:  $\mathcal{S}$      /* a GERM schema
        h      /* highest level of object types in $\mathcal{S}$
        $\Sigma$      /* a set of participation cardinality constraints on $\mathcal{S}$
Output: $\mathcal{T}$      /* a GML schema
Begin
    use Rule 1 to generate the root element of $\mathcal{T}$ ;
    for each object type $O \in \mathcal{S}$ of level 0
        use Rule 2 to transform $O$ ;
        for each attribute $A \in attr(O)$
            use Rule 5 to transform $A$ ;
        endfor
        use Rule 6 to transform the primary key of $O$ ;
    endfor
    for $l := 1$ to $h$
        for each object type $O \in \mathcal{S}$ of level $l$
            use Rule 3 to transform $O$ ;
            use Rule 4 to transform the cardinality constraint defined on $O$ ;
            for each attribute $A \in attr(O)$
                use Rule 5 to transform $A$ ;
            endfor
            use Rule 6 to transform the primary key of $O$ ;
            use Rule 7 to assure referential integrity for $O$ ;
            endfor
        endfor
    endfor
End
```

Example 3. When transforming the GERM schema in our Example 2 using the proposed algorithm we obtain the GML schema depicted in Figure 4. □

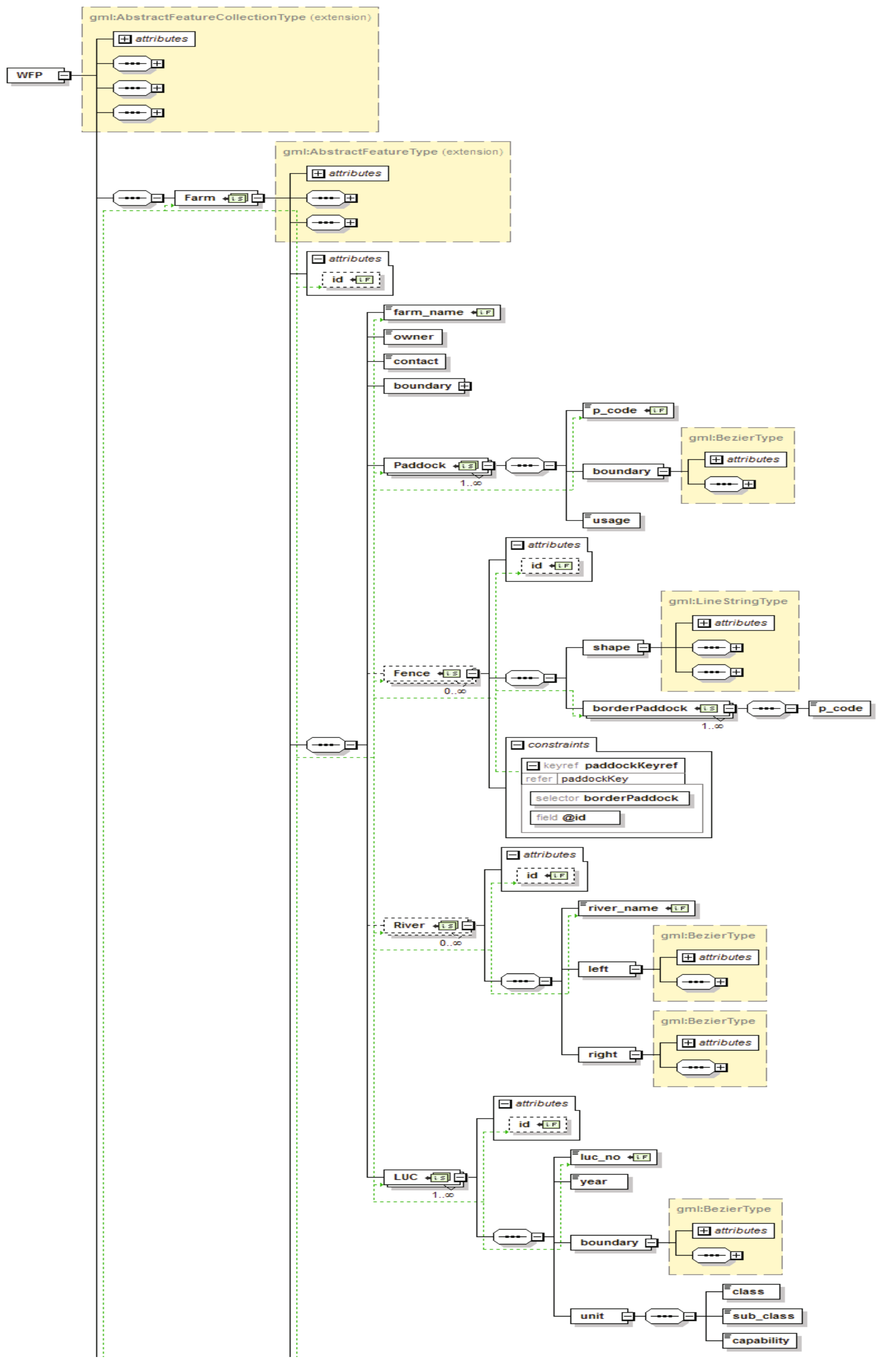


Fig. 4. Example of a GML schema for whole farm plans

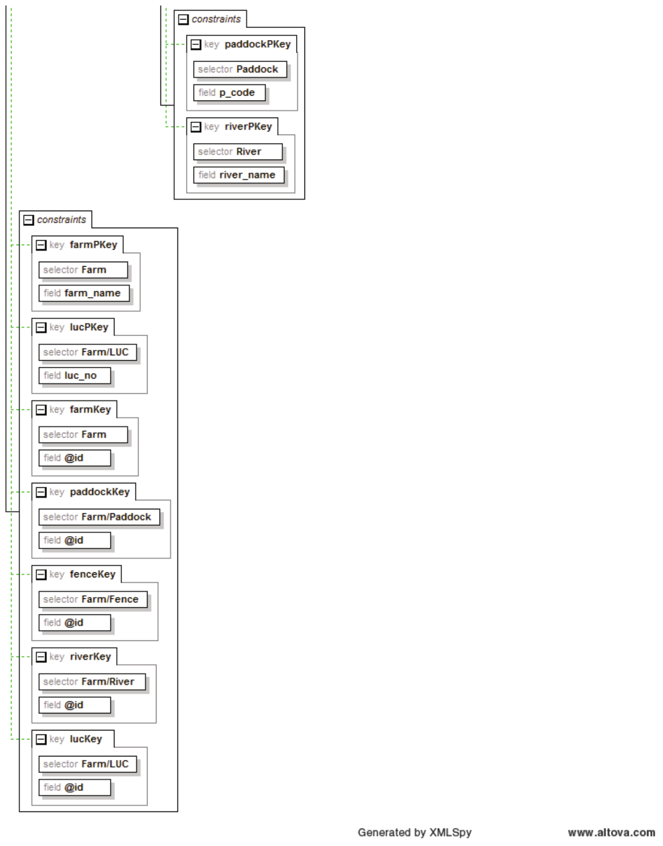


Fig. 4. *(continued)*

5 Conclusion

In this paper we discuss the transformation from geometrically enhanced ER model (GERM) to GML. The transformation from GERM schemas to application schemas in GML presented above is inspired by the general transformation criteria discussed in the literature. Firstly, it preserves semantic information. All object types and their attributes are transformed into elements and subelements. Integrity constrains (such as primary keys, referential integrity, and cardinality constraints) are preserved as far as possible. (For a discussion of the general limitations, see [17].) It is easy to check that the transformation does not create new redundancy. Every object type from the GERM schema is mapped to a unique element hosting all its attributes in the GML schema. The hierarchical structure of the GERM schema is kept as far as possible during the level-wise processing of the transformation algorithm. In fact, relationship types are always nested under their owner component, which has been identified using query frequency information. This approach improves query performance as paths in the GML schema can be exploited as often as possible, thus avoiding expensive join computations.

In earlier work, we have identified GERM as a useful conceptual modelling language for designing geographic applications. In this paper, we have extended our work on GERM by discussing rule-based mappings of GERM schemas to GML which is widely used as an implementation-level format for representing geographic information and sharing across the Internet. We have successfully applied our approach to obtain GML application schemas for whole farm planning in the context of the SLUI initiative.

References

1. AgResearch. Farm plan prototype for SLUI, retrievable online from the New Zealand Association of Resource Management (2005), http://www.nzarm.org.nz/KinrossWholeFarmPlan_A4_200dpi_secure.pdf
2. Al-Kamha, R., Embley, D.W., Liddle, S.W.: Foundational data modeling and schema transformations for XML data engineering. In: UNISCON. LNBIP, vol. 5, pp. 25–36. Springer, Heidelberg (2008)
3. Buneman, P., Davidson, S., Fan, W., Hara, C., Tan, W.: Keys for XML. Computer Networks 39(5), 473–487 (2002)
4. Fornari, M.R., Iochpe, C.: Mapping of conceptual object oriented models to gml. In: International Conference on IADIS WWW/Internet, pp. 444–451 (2002)
5. Franceschet, M., Gubiani, D., Montanari, A., Piazza, C.: From Entity Relationship to XML Schema: A Graph-Theoretic Approach. In: Bellahsène, Z., Hunt, E., Rys, M., Unland, R. (eds.) XSym 2009. LNCS, vol. 5679, pp. 165–179. Springer, Heidelberg (2009)
6. Franceschet, M., Montanari, A., Gubiani, D.: Modeling and validating spatio-temporal conceptual schemas in XML schema. In: International Conference on Database and Expert Systems Application – DEXA, pp. 25–29. IEEE (2007)
7. Frank, A.U.: Map Algebra Extended with Functors for Temporal Data. In: Akoka, J., Liddle, S.W., Song, I.-Y., Bertolotto, M., Comyn-Wattiau, I., van den Heuvel, W.-J., Kolp, M., Trujillo, J., Kop, C., Mayr, H.C. (eds.) ER Workshops 2005. LNCS, vol. 3770, pp. 194–207. Springer, Heidelberg (2005)

8. Hadzilacos, T., Tryfona, N.: An extended entity-relationship model for geographic applications. SIGMOD Record 26(3), 24–29 (1997)
9. Hartmann, S., Köhler, H., Link, S., Trinh, T., Wang, J.: On the Notion of an XML Key. In: Schewe, K.-D., Thalheim, B. (eds.) SDKB 2008. LNCS, vol. 4925, pp. 103–112. Springer, Heidelberg (2008)
10. Hartmann, S., Link, S.: Efficient reasoning about a robust XML key fragment. ACM Transactions on Database Systems 34(2) (2009)
11. Hora, A.C., Davis Jr., C.A., Moro, M.M.: Generating XML/GML schemas from geographic conceptual schemas. In: Foundations of Data Management, Alberto Mendelzon International Workshop – AMW (2010)
12. Hull, R., King, R.: Semantic database modeling: Survey, applications, and research issues. ACM Computing Surveys 19(3), 201–260 (1987)
13. Ishikawa, Y., Kitagawa, H.: Source Description-Based Approach for the Modeling of Spatial Information Integration. In: Kunii, H.S., Jajodia, S., Sølvberg, A. (eds.) ER 2001. LNCS, vol. 2224, pp. 41–55. Springer, Heidelberg (2001)
14. Kleiner, C., Lipeck, U.W.: Automatic generation of XML DTDs from conceptual database schemas. In: GI Jahrestagung, pp. 396–405 (2001)
15. Krumbein, T., Kudrass, T.: Rule-based generation of XML schemas from UML class diagrams. In: Berliner XML Tage, pp. 213–227 (2003)
16. Li, Y., Lu, J., Guan, J., Fan, M., Haggag, A., Yahagi, T.: GML topology data storage schema design. Journal of Advanced Computational Intelligence and Intelligent Informatics 11(6), 701–708 (2007)
17. Link, S., Trinh, T.: Know your limits: Enhanced XML modeling with cardinality constraints. In: Conceptual Modeling – ER Tutorials. CRPIT, vol. 83, pp. 19–30. Australian Computer Society (2007)
18. Liu, C., Li, J.: Designing Quality XML Schemas from E-R Diagrams. In: Yu, J.X., Kitsuregawa, M., Leong, H.-V. (eds.) WAIM 2006. LNCS, vol. 4016, pp. 508–519. Springer, Heidelberg (2006)
19. Ma, H.: A geometrically enhanced conceptual model and query language. Journal of Universal Computer Science 16(20), 2986–3015 (2010)
20. Ma, H., Schewe, K.-D., Thalheim, B.: Geometrically Enhanced Conceptual Modelling. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., de Oliveira, J.P.M. (eds.) ER 2009. LNCS, vol. 5829, pp. 219–233. Springer, Heidelberg (2009)
21. Mackay, A.: Specifications of whole farm plans as a tool for affecting land use change to reduce risk to extreme climatic events. AgResearch (2007)
22. OpenGIS Consortium. OGC Standards and Specifications
23. Pigozzo, P., Quintarelli, E.: An algorithm for generating XML schemas from ER schemas. In: Advanced Database Systems, Italian Symposium – SEBD, pp. 192–199 (2005)
24. Price, R., Tryfona, N., Jensen, C.S.: Modeling Topological Constraints in Spatial Part-Whole Relationships. In: Kunii, H.S., Jajodia, S., Sølvberg, A. (eds.) ER 2001. LNCS, vol. 2224, pp. 27–40. Springer, Heidelberg (2001)
25. Shekhar, S., Coyle, M., Liu, D.-R., Goyal, B., Sarkar, S.: Data models in geographic information systems. Communications of the ACM 40(4), 103–111 (1997)
26. Shekhar, S., Vatsavai, R.R., Chawla, S., Burk, T.E.: Spatial Pictogram Enhanced Conceptual Data Models and Their Translation to Logical Data Models. In: Agouris, P., Stefanidis, A. (eds.) ISD 1999. LNCS, vol. 1737, pp. 77–104. Springer, Heidelberg (1999)
27. Thalheim, B.: Entity Relationship Modeling - Foundations of Database Technology. Springer, Heidelberg (2000)

Extensional Logic of Hyperintensions

Marie Duží

VSB-Technical University Ostrava, Czech Republic
marie.duzi@vsb.cz

Abstract. In this paper I describe an extensional logic of hyperintensions, *viz.* Tichý's Transparent Intensional Logic (TIL). TIL preserves transparency and compositionality in all kinds of context, and validates quantifying into all contexts, including intensional and hyperintensional ones. The availability of an extensional logic of hyperintensions defies the received view that an intensional (let alone hyperintensional) logic is one that fails to validate transparency, compositionality, and quantifying-in. The main features of our logic are that the senses and denotations of (non-indexical) terms and expressions remain invariant across contexts and that our ramified type theory enables quantification over any logical objects of any order. The syntax of TIL is the typed lambda calculus; its semantics is based on a procedural redefinition of, inter alia, functional abstraction and application. The only two non-standard features are a hyperintension (called *Trivialization*) that presents other hyperintensions and a four-place substitution function (called *Sub*) defined over hyperintensions.

Keywords: Quantifying-in, extensional/intensional/hyperintensional context, transparency, ramified type theory, transparent intensional logic, extensional logic of hyperintensions.

1 Introduction

In this paper I introduce basic fundamentals of an extensional logic of hyperintensions developed within *procedural semantics* of Transparent Intensional Logic (TIL). Only an extensional logic will validate extensional principles like the rule of existential generalization, hence only an extensional logic of hyperintensions will stand a chance of validating quantifying-in. The cornerstone of TIL approach is that we avail ourselves of rich ontology organised into an infinite bi-dimensional hierarchy of types. We assign to terms and expressions occurring in hyperintensional contexts the very same meaning that we assign to those very same terms and expressions when occurring in intensional and extensional contexts. As a result of this top-down approach, the extensional logical rules apply indiscriminately to all contexts. The upside of our top-down approach is that referential transparency and compositionality of meaning are preserved throughout, together with semantic innocence, since we have no recourse to reference shift. At no point do we invoke contextualist epicycles to somehow create a second semantics for 'non-extensional' contexts. The perceived downside would be that we revise the prevalent extensionalist semantic theory of terms and expressions, in that we universalize Frege's semantics earmarked for *Sinn*-sensitive contexts to all contexts. Be that as it may, it is strength of our solution that it is emphatically not tailor-made specifically for validating extensional principles. Instead it is just yet another application of a large-scale background theory and our solutions are principled and not *ad hoc*.

The rest of the paper is organised as follows. In Section 2 I briefly summarise the history of the development of logical semantics from Frege *via* so-called syntactic and model-theoretic turn up to procedural or algorithmic turn in semantics. Section 3 introduces the core of TIL so that to describe logical machinery needed in the main Section 4 where basic principles of extensional logic of hyperintensions are introduced.

2 Historical Background

The second half of the last century can be characterized as a linguistic turn in semantics. We were developing systems of particular logics which are characterized by a language with a precisely defined syntax and a model set-theoretic semantics. The main goal of building such a system is to find a subset of sentences of the language, axioms of the theory, which characterize a given area under scrutiny and apply proper rules of inference in order to mechanically derive consequences of the axioms. If the system has a model, then it is consistent, and all we are interested in is manipulating symbols. Therefore, *linguistic or syntactic turn.*

Says David Kaplan:

> During the Golden Age of Pure Semantics we were developing a nice homogenous theory, with language, meanings, and entities of the world each properly segregated and related one to another in rather smooth and comfortable ways. This development probably came to its peak in *Carnap*'s Meaning and Necessity (1947). Each *designator* has both an intension and an extension. Sentences have truth-values as extensions and propositions as intensions, predicates have classes as extensions and properties as intensions, terms have individuals as extensions and individual concepts as intensions …. The intension of a compound is a function of the intensions of the parts and similarly the extension (except when intensional operators appear). There is great beauty and power in this theory. But there remained some nagging doubts: proper names, demonstratives, and quantification into intensional contexts. (1990, pp. 13-14)

The mainstream in this direction was *Possible World Semantics* (PWS). Kripke characterizes this semantics as follows:

> We define a ***proposition*** (…) as a *mapping* whose domain is ***K*** [a *logical space of possible worlds*] and whose range is the set {T, F}. (Intuitively, a proposition is something that can be *true* or *false* in each world; and (…) we identify propositions that are *strictly equivalent*, i.e. have the same truth-value in each world. (…) Notice that each proposition determines a unique set of worlds (the set of all worlds mapped into T), and that conversely each set of worlds determines a proposition (its 'characteristic function'). Thus a *proposition* could just as well have been defined simply as *a subset of K.* (1963, §5.3)

Possible-world intensions are extensionally individuated:

$$\forall fg\ (\forall w\ (fw = gw) \supset f = g)$$

Thus Possible-world semantics is an *extensional logic of intensions* and the *model-theoretic* (hence set-theoretic) *theory of modalities*. Yet its individuation of meaning is too crude, up to equivalence. Notoriously well-known problem that is a test for an expressive power of a given semantic theory is the analysis of belief sentences. Carnap in (1947) says that modal sentences like "It is necessary that *P*" are intensional with respect to the clause *P*. However, sentences about belief like "John believes that *P*" are neither intensional nor extensional with respect to *P*. Carnap does not accept the proposal of C.I. Lewis (1918) to solve the problem by means of intensions, extensions and comprehension (that is including impossible possible things). He objects against Meinong's classification of things as impossible, possible and actual, possible and non-actual, because existence is not a property of things but of intensions.[1]

In order to solve the problem of belief sentences, Carnap proposed the method of *intensional isomorphism* and defined inductively the relation of intensional isomorphism on the set of sentences. Roughly, two sentences *S* and *P* are intensionally isomorphic if they are L-equivalent and each designator (either simple or composed) that is a constituent of *S* is L-equivalent to the respective designator of *P*. Thus sentences *S* and *P* have the same intensional structure if they are composed *in the same way* from designators with the same intensions.

Carnap did not accept sentencialism and tried to define a stronger relation between expressions that might rightly calibrate the identity of meaning (i.e. *synonymy*) than L-equivalence. He applied his notion of an intensional structure in the analysis of sentences about belief and the paradox of analysis. In my opinion, all these Carnap's tenets and philosophical desiderata are plausible and it might seem that he really succeeded in defining the subject of beliefs, knowledge, convictions, etc. Moreover, his definition is independent of the language and syntactic structure in which this subject is encoded. So far so good; yet Carnap's method has been criticized in particular by Alonzo Church (1954). Church's argument is based on two principles. First, it is Carnap's principle of tolerance (which itself is, of course, desirable), and second, which is less desirable, this principle makes it possible to introduce into a language syntactically simple expressions as definitional abbreviations of semantically complex expressions (for instance, in English 'fortnight' standing for 'a period of fourteen days').

Thus we can introduce into a language *primitive* symbols *P* and *Q* in this way:

> *P* is the set of natural numbers that are less than the number 3.
> *Q* is the set of natural numbers *n* for which there are natural numbers *x*, *y*, *z* such that $x^n + y^n = z^n$.

[1] See Corazon, R.: `http://www.formalontology.com/pdf/meinonga.pdf`; parts of these paragraphs draw on material published in Jespersen (2010).

But then *P* and *Q* are L-equivalent (because they denote the same set of numbers) and also intensionally isomorphic because they have no other constituent designators but themselves. Yet it is easy to believe that $\exists n\ (Qn \wedge \neg Pn)$ without believing that $\exists n\ (Pn \wedge \neg Pn)$.[2] Church proposes a *synonymous isomorphism*: all the mutually corresponding designators must be not only L-equivalent but also synonymous, where the synonymy of syntactically simple designators must be *postulated* as a semantic base of a language. We can postulate any convention for introducing these synonymous abbreviations, but as soon as we postulate the meaning of a constant it becomes valid and cannot be changed by another convention.

Going back to the history we encounter Frege who was (to the best of my knowledge) the first who were developing formal semantics. In (1892) Frege introduced the well-known semantic schema assigning to expressions their *sense* (*Sinn*) and *denotation* (*Bedeutung*). Wishing to save compositionality, Frege made the semantics of an expression depend on the linguistic context in which it is embedded. According to Frege an expression names its *Bedeutung* (extension) in ordinary contexts and *Sinn* (intension) in oblique contexts.[3] The price is very high, indeed. No expression can denote an object, unless a particular kind of context is provided. Yet such a solution is far from being natural. There are cases of real ambiguity, witness homonymous expressions. Which of the denotations is relevant in such cases (e.g., 'is a bank') can be detected by a particular context (cf. "A bank was robbed" vs. "A woman walks along the banks of the Dnepr"), but would anybody say that 'The author of *Waverley*' were another such case of homonymy? Hardly; unless, of course, their intuitions had been warped by Fregean contextualism. Furthermore, expressions can be embedded within other expressions to various degrees; consider the sentence

"Charles knows that Tom believes that the author of *Waverley* is a poet."

The expression 'The author of *Waverley*' should now denote the 'normal' sense of the 'normal sense' of itself. Adding still further layers of embedding sets off an infinite hierarchy of senses, which is to say that 'The author of *Waverley*' has the potential of being infinitely ambiguous. This seems plain wrong, and is first and foremost an awkward artefact of Fregean semantics.[4] For these reasons Carnap also criticised the 'naming method' (now we would say the denotational semantics), because then necessarily we multiply the names *ad infinitum*, and we end up with the antinomy of naming. For Carnap extension is not a matter of *logical semantics* because it is a matter of factual knowledge. Prior for the meaning is an *intension* that uniquely determines the extension (if any), but not *vice versa*, and is independent of contingent facts.

There are many other objections against the set-theoretical denotational semantics. The arguments could be summarised as follows. First, one and the same denotation can be produced by infinitely many "senses". Second, in the "flat" (set-theoretic) denotation there is no trace of particular constituents of the meaning (sense) of an

[2] For the proof of Fermat's theorem is difficult to discover (written in 1954).

[3] There is another defect in Frege's semantics; extension of a sentence is its truth-value. Yet in case of empirical sentences, the truth-value depends on contingent facts, which is not a matter of *logical* semantics. See also Klement (2002).

[4] See also Duží & Materna (2010), or Duží, Jespersen & Materna (2010, §1.5).

expression. Third, knowing the meaning of an expression amounts for understanding it. And we do understand many expressions without knowing the respective object denoted by the expression. And we do understand even those expressions that do not denote anything, and thus according to the denotational set-theoretic semantics they are meaningless. For instance, mathematicians had to understand 'the greatest prime' prior to proving that there is no such number. They understood the 'instruction' what to do prior to proving that this is a blind alley yielding nothing. And finally, in case of empirical expressions which always have a denotation (PWS intension), possible-world-semantics tells us that knowing the meaning amounts for knowing this infinite uncountable mapping from possible worlds and times to the set of objects of a respective type. But then we would be empirically omniscient! No human being with limited capacities can know such an actual infinity.

Thus since the late 60-s of the last century many logicians strived for *hyperintensional semantics* and *structured meanings.*[5] The structured character of meaning was urged by David Lewis in (1972), where non-structured intensions are generated by finite, ordered trees. This idea of 'tree-like' meanings obviously influenced George Bealer's idea of 'intensions of the second kind' in his (1982). The idea of structured meanings was propagated also by M.J. Cresswell in (1975) and (1985). He defines structured meanings as ordered *n*-tuples. That this is far from being a satisfactory solution is shown in Tichý (1994) and Jespersen (2003). In brief, these tuples are again set-theoretic entities structured at most from a mereological point of view, by having elements or parts (though one balks at calling elements 'parts', since sets, including tuples, are not complexes). Besides, tuples are of the wrong making to serve as truth-bearers and objects of attitudes, since a tuple cannot be true or be known, hoped, etc., to be true. Simply, tuples are 'flat' from the procedural or algorithmic point of view. The *way* of combining particular parts together is missing here.

In 1994 Moschovakis comes with an idea of *meaning as algorithm*. The meaning of a term *A* is "an (abstract, idealized, not necessarily implementable) algorithm which computes the denotation of *A*" (2006, p. 27; see also 1994).[6] Moschovakis outlines his conception thus:

> The starting point … [is] the insight that a correct understanding of programming languages should explain the relation between a program and the algorithm it expresses, so that the basic interpretation scheme for a programming language is of the form
>
> $$\text{program } P \rightarrow \text{algorithm}(P) \rightarrow \text{den}(P).$$
>
> It is not hard to work out the mathematical theory of a suitably abstract notion of algorithm which makes this work; and once this is done, then it is hard to miss the similarity of the above schema with the basic Fregean scheme for the interpretation of a natural language,
>
> $$\text{term } A \rightarrow \text{meaning}(A) \rightarrow \text{den}(A).$$

[5] See also Fox and Lappin (2001).

[6] Moschovakis' notion of algorithm borders on being too permissive, since algorithms are normally understood to be effective. (See Cleland (2002) for discussion.)

> This suggested at least a formal analogy between algorithms and meanings which seemed worth investigating, and proved after some work to be more than formal: when we view natural language with a programmer's eye, it seems almost obvious that we can represent the meaning of a term *A* by the algorithm which is expressed by *A* and which computes its denotation. (2006, p. 42)

Yet much earlier, in (1968) and (1969) Pavel Tichý formulated the idea of *procedural semantics.* Thus, for instance, a sentence encodes an *instruction* how in any possible world at any time to execute the abstract *procedure* expressed by the sentence as its meaning, i.e., to evaluate the truth-conditions of the sentence. He developed a logical framework known today as *Transparent Intensional Logic* (TIL). In modern jargon, TIL belongs to the paradigm of *structured meaning*. However, Tichý does not reduce structure to set-theoretic sequences, as do Kaplan and Cresswell. Nor does Tichý fail to explain how the sense of a molecular term is determined by the senses of its atoms and their syntactic arrangement, as Moschovakis objects to 'structural' approaches in (2006, p. 27).

3 Foundations of TIL

From the formal point of view, TIL is a hyperintensional, partial typed λ-calculus.[7] A main feature of the λ-calculus is its ability to systematically distinguish between functions and functional values. An additional feature of TIL is its ability to systematically distinguish between functions and modes of presentation of functions and modes of presentation of functional values. The TIL operation known as *Closure* is the very *procedure* of presenting or forming or obtaining or *constructing* a function; the TIL operation known as *Composition* is the very *procedure* of *constructing* the value (if any) of a function at an argument. Compositions and Closures are both multiple-step procedures, or *constructions*, that operate on input provided by two one-step constructions, which figure as sub-procedures of Compositions and Closures, namely *variables* and so-called *Trivializations*. Characters such as '*x*', '*y*' '*z*' are words denoting variables, which construct the respective values that an assignment function has accorded to them. The linguistic counterpart of a Trivialization is a constant term always picking out the same object. An analogy from programming languages might be helpful. The Trivialization of an object *X*, whatever *X* may be, and its use are comparable to a *fixed pointer* to *X* and the *dereference* of the pointer. In order to operate on *X*, *X* needs to be grabbed first. Trivialization is such a one-step grabbing mechanism. Similarly, in order to talk about China (in non-demonstrative and non-indexical English discourse), we need to name China, most simply by using the constant 'China'. Trivialization is important in what follows, because in order to substitute one sub-construction for another inside a construction it is crucial to be able to grab those three individual constructions.

The logical core of TIL is its notion of *construction* and its *type hierarchy*, which divides into a ramified type theory and a simple type theory. The ramified type hierarchy organizes all higher-order objects, which are all constructions, as well as all

[7] For details on TIL see in particular Tichý (1988, 2004) and Duží et al (2010).

functions with domain or range in constructions. The simple type hierarchy organizes first-order objects, which are non-constructions like extensions (individuals, numbers, sets, etc.), possible-world intensions (functions from possible worlds) and their arguments and values, including those values whose values are themselves intensions (like *Einstein's favourite proposition*). The relevant definitions decompose into three parts. Firstly, simple types of order 1 are defined by Definition 1. Secondly, constructions of order n, and thirdly, types of order $n + 1$.

Definition 1 (*types of order 1*). Let B be a *base*, where a base is a collection of pair-wise disjoint, non-empty sets. Then:

(i) Every member of B is an elementary *type of order 1 over B*.
(ii) Let α, β_1, ..., β_m ($m > 0$) be types of order 1 over B. Then the collection ($\alpha\ \beta_1 \ldots \beta_m$) of all m-ary partial mappings from $\beta_1 \times \ldots \times \beta_m$ into α is a functional *type of* order *1 over B*.
(iii) Nothing is a *type of order 1 over B* unless it so follows from (i) and (ii).

Remark. For the purposes of natural-language analysis, we are currently assuming the following base of ground types, which is part of the ontological commitments of TIL:

o:	the set of truth-values {**T, F**};
ι:	the set of individuals (constant universe of discourse);
τ:	the set of real numbers (doubling as temporal continuum);
ω:	the set of logically possible worlds (logical space).

Definition 2 (*construction*)

(i) The *variable x* is a *construction* that constructs an object O of the respective type dependently on a valuation v: x v-constructs O.
(ii) *Trivialization*: Where X is an object whatsoever (an extension, an intension or a *construction*), ${}^0X$ is the *construction Trivialization.* It constructs X without any change.
(iii) The *Composition* $[X\ Y_1 \ldots Y_m]$ is the following *construction.* If X *v-constructs* a function f of type ($\alpha\ \beta_1 \ldots \beta_m$), and $Y_1, \ldots, Y_m$ *v-construct* entities $B_1, \ldots, B_m$ of types $\beta_1, \ldots, \beta_m$, respectively, then the *Composition* $[X\ Y_1 \ldots Y_m]$ *v-constructs* the value (an entity, if any, of type α) of f on the tuple argument $\langle B_1, \ldots, B_m \rangle$. Otherwise the *Composition* $[X\ Y_1 \ldots Y_m]$ does not *v-construct* anything and so is *v-improper.*
(iv) The *Closure* $[\lambda x_1 \ldots x_m\ Y]$ is the following *construction.* Let $x_1, x_2, \ldots, x_m$ be pair-wise distinct variables v-constructing entities of types $\beta_1, \ldots, \beta_m$ and Y a construction v-constructing an α-entity. Then $[\lambda x_1 \ldots x_m\ Y]$ is the *construction* λ-*Closure* (or *Closure*). It *v-constructs* the following function f of the type (α $\beta_1 \ldots \beta_m$). Let $v(B_1/x_1,\ldots,B_m/x_m)$ be a valuation identical with v at least up to assigning objects $B_1/\beta_1, \ldots, B_m/\beta_m$ to variables $x_1, \ldots, x_m$. If Y is $v(B_1/x_1, \ldots, B_m/x_m)$-improper (see iii), then f is undefined on $\langle B_1, \ldots, B_m \rangle$. Otherwise the value of f on $\langle B_1, \ldots, B_m \rangle$ is the α-entity $v(B_1/x_1,\ldots,B_m/x_m)$-constructed by Y.

(v) The *Single Execution* ^1X is the *construction* that either v-constructs the entity v-constructed by X or, if X is not itself a construction or X is v-improper, ^1X is v-improper.

(vi) The *Double Execution* ^2X is the following *construction*. Where X is any entity, the *Double Execution* ^2X is *v-improper* (yielding nothing relative to v) if X is not itself a construction, or if X does not v-construct a construction, or if X v-constructs a v-improper construction. Otherwise, let X v-construct a construction Y and Y v-construct an entity Z: then ^2X v-constructs Z.

(vii) Nothing is a *construction*, unless it so follows from (i) through (vi).

Definition 3 (*ramified hierarchy of types*)
$\mathbf{T_1}$ *(types of order 1)*. See Definition 1.

$\mathbf{C}_n$ *(constructions of order n)*

i) Let x be a variable ranging over a type of order n. Then x is a *construction of order n over B*.
ii) Let X be a member of a type of order n. Then ^0X, ^1X, ^2X are *constructions of order n over B*.
iii) Let $X, X_1,..., X_m$ ($m > 0$) be constructions of order n over B. Then $[X\, X_1... X_m]$ is a *construction of order n over B*.
iv) Let $x_1,...x_m, X$ ($m > 0$) be constructions of order n over B. Then $[\lambda x_1...x_m\, X]$ is a *construction of order n over B*.
v) Nothing is a *construction of order n over B* unless it so follows from $\mathbf{C}_n$ (i)-(iv).

$\mathbf{T}_{n+1}$ *(types of order n + 1)* Let $*_n$ be the collection of all constructions of order n over B. Then

i) $*_n$ and every type of order n are *types of order n + 1*.
ii) If $0 < m$ and $\alpha, \beta_1,...,\beta_m$ are types of order $n + 1$ over B, then $(\alpha\, \beta_1\, ...\, \beta_m)$ (see T_1 ii)) is a *type of order* $n + 1$ *over B*.
iii) Nothing is a *type of order n + 1 over B* unless it so follows from (i) and (ii).

Examples of mathematical constructions (now we work over the base $\{o, \nu\}$, where ν is the type of natural numbers; all the variables range over this type).

a) The function + is not a construction. It is a mapping of type $(\nu\nu\nu)$, i.e., the set of triples, the first two members of which are natural numbers, while the third member is their sum. The simplest construction of this mapping is its Trivialization $^0+$. (See Definition 2, ii)). This function + can be constructed by infinitely many equivalent, yet distinct constructions; for instance, the following Closures are equivalent by constructing the same mapping: $\lambda xy\, [^0+\, x\, y]$, $\lambda yx\, [^0+\, x\, y]$, $\lambda xz\, [^0+\, x\, z]$, $\lambda xy\, [^0+\, [^0-\, [^0+\, x\, y]\, y]\, y]$. (See Definition 2, iii), iv)).
b) The Composition $[^0+\ ^02\ ^05]$ constructs the number 7, i.e., the value of the function + (constructed by $^0+$) at the argument $\langle 2, 5\rangle$ constructed by ^{02} and ^{05}. (See Definition 2, iii.)) Note that the numbers 2, 5 are not constituents of this Composition, nor is the function +. Instead, the Trivialisations $^0+$, ^{02}, ^{05} are the constituents of the Composition $[^0+\ ^02\ ^05]$.

c) The Composition $[{}^0{+}\ x\ {}^01]$ v-constructs the successor of any number x; again, the number 1 is not a constituent of this Composition. Instead, the Trivialisation ^{01} is a constituent. The other two constituents are ${}^0{+}$, x.

d) The Closure $\lambda x\ [{}^0{+}\ x\ {}^01]$ constructs the successor function of type $(\nu\nu)$. (See Definition 2, iv.)) The successor function can be constructed by infinitely many constructions, the simplest one of which is the Trivialisation of the function *Succ*/$(\nu\nu)$: ${}^0Succ$. Thus $\lambda x\ [{}^0{+}\ x\ {}^01]$ and ${}^0Succ$ are equivalent by constructing the same function. Yet the Trivialization ${}^0Succ$ is not a finitary, executable procedure. It is a one-step procedure producing an *(infinite) mapping* as its product. On the other hand, the Closure $\lambda x\ [{}^0{+}\ x\ {}^01]$ is an easily executable procedure. The instruction to execute this procedure can be decomposed into the following steps: *Take any number x and the number 1; apply the function + to the couple of numbers obtained in the previous step; abstract from the number x.* The Composition of this closure with ^{05}, i.e., $[\lambda x\ [{}^0{+}\ x\ {}^01]\ {}^05]$, constructs the number 6. (See Definition 2, iii.))

e) The Composition $[{}^0{:}\ x\ {}^00]$, where $:/(\nu\nu\nu)$ is the division function, does not v-construct anything for any valuation of x; it is *v-improper* for any valuation v. (See Definition 2, iii.)) We will say 'improper', for short.

f) The closure $\lambda x\ [{}^0{:}\ x\ {}^00]$ is not improper, as it constructs something, even though it is only a *degenerate function*, *viz.* one undefined at all its arguments. (See Definition 2, iv.))

g) Compositions $[{}^0\exists\lambda x\ [{}^0{>}\ x\ {}^05]]$, $[{}^0\forall\lambda x\ [{}^0{>}\ x\ {}^05]]$ construct the truth-value **T** and **F**, respectively, because the class of natural numbers greater than 5 constructed by the Closure $\lambda x\ [{}^0{>}\ x\ {}^05]$ is not empty, but is not the whole type ν.

h) If $I^\tau/(\tau(o\tau))$ is a singularizer, then the following construction (the meaning of 'the greatest prime') is improper:
$[{}^0I^\tau\ \lambda x\ [{}^0\wedge\ [{}^0Prime\ x]\ [{}^0\forall\lambda y\ [{}^0\supset\ [{}^0Prime\ y]\ [{}^0\geq\ x\ y]]]]]$, or for short, $[{}^0I\ \lambda x\ [[{}^0Prime\ x] \wedge \forall y\ [[{}^0Prime\ y] \supset [{}^0\geq x\ y]]]]$.

Empirical languages incorporate an element of *contingency* that non-empirical ones lack. Empirical expressions denote *empirical conditions* that may, or may not, be satisfied at some empirical index of evaluation. Non-empirical languages have no need for an additional category of expressions for empirical conditions. We model these empirical conditions as *possible-world intensions.* Intensions are entities of type $(\beta\omega)$: mappings from possible worlds to an arbitrary type β. The type β is frequently the type of the *chronology* of α-objects, i.e., a mapping of type $(\alpha\tau)$. Thus α-intensions are frequently functions of type $((\alpha\tau)\omega)$, abbreviated as '$\alpha_{\tau\omega}$'. We shall typically say that an index of evaluation is a world/time pair $\langle w, t\rangle$. *Extensional entities* are entities of a type α where $\alpha \neq (\beta\omega)$ for any type β.

Examples of frequently used intensions are: *propositions* of type $o_{\tau\omega}$ (denoted by non-indexical sentences), *properties* of individuals of type $(o\iota)_{\tau\omega}$ (denoted by predicates or nouns like 'being happy', 'being a mathematician'), binary *relations-in-intension* between individuals of type $(o\iota\iota)_{\tau\omega}$ (usually denoted by verbs like 'admire', 'kick'), individual *offices* or *roles* of type $\iota_{\tau\omega}$ (denoted by definite descriptions like 'pope', 'the president of CR', 'the first man to run 100 m under 9 s'), binary relations-in-intension between individuals and hyperintensions of type $(\omicron\iota*_n)_{\tau\omega}$ (denoted by attitudinal verbs like 'believe', 'know', 'calculate', etc.).

The method of explicit intensionalization and temporalization encodes constructions of possible-world intensions directly in the logical syntax. Where w ranges over ω and t over τ, the following logical form essentially characterizes the logical syntax of empirical language: $\lambda w \lambda t\ [\ldots w \ldots. t \ldots]$. If the Composition $[\ldots w \ldots. t \ldots]$ v-constructs an entity of type α, then the Closure itself constructs a function of type $((\alpha\tau)\omega)$, or $\alpha_{\tau\omega}$ for short, i.e. an α-intension.

Logical objects like *truth-functions* and *quantifiers* are extensional: $\wedge$ (conjunction), $\vee$ (disjunction) and $\supset$ (implication) are of type $(\omicron\omicron\omicron)$, and $\neg$ (negation) of type $(\omicron\omicron)$. *Quantifiers* $\forall^\alpha$, $\exists^\alpha$ are type-theoretically polymorphous total functions of type $(\omicron(\omicron\alpha))$, for an arbitrary type α, defined as follows. The *universal quantifier* $\forall^\alpha$ is a function that associates a class A of α-elements with **T** if A contains all elements of the type α, otherwise with **F**. The *existential quantifier* $\exists^\alpha$ is a function that associates a class A of α-elements with **T** if A is a non-empty class, otherwise with **F**.

Notational conventions: Below all type indications will be provided outside the formulae in order not to clutter the notation. Furthermore, 'X/α' means that an object X is (a member) of type α. '$X \rightarrow_v \alpha$' means that the type of the object v-constructed by X is α. We write '$X \rightarrow \alpha$' if what is v-constructed does not depend on a valuation v. Throughout, it holds that the variables $w \rightarrow_v \omega$ and $t \rightarrow_v \tau$. If $C \rightarrow_v \alpha_{\tau\omega}$ then the frequently used Composition $[[C\ w]\ t]$, which is the extensionalization of the α-intension v-constructed by C, will be encoded as 'C_{wt}'. When applying truth-functions, identities $=^\alpha/(\omicron\alpha\alpha)$, arithmetic operations and relations $>^\alpha$, $<^\alpha$, we will often use an infix notation without Trivialization and without indicating the type of a function. Instead of '$[^0\exists^\alpha\ \lambda x\ B]$', '$[^0\forall^\alpha\ \lambda x\ B]$' ($x \rightarrow_v \alpha$; $B \rightarrow_v \omicron$) we will often write '$\exists x B$', '$\forall x B$'. Thus, for instance, $[^0\forall^\tau\lambda x\ [^0\supset [^0=^\tau x\ ^00]\ [^0\forall^\tau\lambda y\ [^0=^\tau [^0+ x\ y]\ y]]]]$ will be encoded as $\forall x\ [[x = {}^00] \supset \forall y\ [[x + y] = y]]$.

The analysis of an expression consists in discovering the construction (meaning) encoded by the expression. To this end we apply a *method of analysis* that consists of three steps:

1) *Type-theoretical analysis*, i.e., assigning types to the objects that receive mention in the analysed sentence.
2) *Synthesis*, i.e., combining the constructions of the objects *ad* (1) in order to construct the proposition of type $\omicron_{\tau\omega}$ denoted by the whole sentence.
3) *Type-Theoretical checking.*

To illustrate the method, let us analyse the sentence "Tom is seeking a vacant parking place".

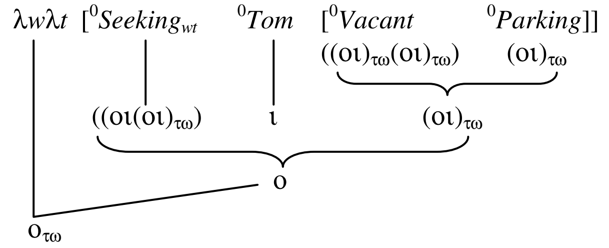
Ad (1) types: *Tom*/ι; *Seeking*/$(\omicron\iota(\omicron\iota)_{\tau\omega})_{\tau\omega}$: the relation-in-intension of an individual to a property the instance of which the individual wants to find; *Parking (Place)*/$(\omicron\iota)_{\tau\omega}$: a property of individuals; *Vacant*/$((\omicron\iota)_{\tau\omega}(\omicron\iota)_{\tau\omega})$: a property modifier. The whole sentence denotes a proposition, i.e. an object of type $\omicron_{\tau\omega}$.

Ad (2). We aim at *literal analysis* of the sentence. Thus semantically simple expressions are analysed as Trivializations of the objects they denote: ^{0}*Tom*, ^{0}*Seek*,

^{0}*Parking*, ^{0}*Vacant*. Now we need to apply the relation *Seeking* to *Tom* and the property of being a vacant parking place. This property is constructed by Composition [^{0}*Vacant* ^{0}*Parking*]. However, the relation-in-intension is not a type-theoretically proper object to be applied to an individual and a property; it must be extensionalised first: $[[{}^0\mathit{Seeking}\ w]\ t] \rightarrow (o\iota(o\iota)_{\tau\omega})$, or $^0\mathit{Seeking}_{wt}$ for short. The Composition $[{}^0\mathit{Seeking}_{wt}\ {}^0\mathit{Tom}\ [{}^0\mathit{Vacant}\ {}^0\mathit{Parking}]] \rightarrow_v o$ *v*-constructs the truth-value **T** or **F** according as in a given world *w* and time *t* of evaluation Tom is seeking a vacant parking or not. In order to construct the proposition denoted by the sentence, we must now abstract over the values of *w* and *t*. Thus we have:

$$\lambda w \lambda t\ [{}^0\mathit{Seeking}_{wt}\ {}^0\mathit{Tom}\ [{}^0\mathit{Vacant}\ {}^0\mathit{Parking}]] \rightarrow o_{\tau\omega}$$

Ad (3) Finally we perform the type-theoretical control in order to check whether particular constituents are combined in compliance with type constraints.



The role of Trivialization in agents' communication can be described as follows. Trivialized entities are the *primitive concepts* that an agent should have in its ontology. If it is not so, the agent must ask its fellow-agents in order to learn a new compound concept. *Compound concepts* are Closures or closed Compositions.

The empirical parameters *w* (the modal one) and *t* (the temporal one) serve as instructions to execute an empirical search for an actual value of a given intension. To this end an agent can consult its own knowledge base or ask the other agents for the facts recorded in their knowledge bases, or even use its own 'senses' to obtain facts from their environment.

To summarize, our neo-Fregean semantic schema, which applies to all contexts, is this:

$$\begin{tikzcd}
\mathit{Expression} \arrow[d, "\mathit{expresses}"] \arrow[dd, dotted, bend right=60, "\mathit{denotes}"'] \\
\mathit{Construction} \arrow[d, dotted, "\mathit{constructs}"] \\
\mathit{Denotation}
\end{tikzcd}$$

The most important relation in this schema is between an expression and its meaning (a construction). We can investigate *a priori* what (if anything) a construction constructs and what is entailed by it. Once a construction is explicitly given as a result of logical analysis, the entity (if any) it constructs is already implicitly given, whereas it requires inquiry *a posteriori* to establish the reference of an empirical term at a given world/time pair. As a limiting case, the logical analysis may reveal that the construction fails to construct anything because it is improper. And if the construction is not improper, the denotation can be either a first-order object (i.e. a non-construction) or a lower-order construction. Intensional constructions (constructions of objects of type $(\beta\omega)$) are always proper, since they always construct an intension (including degenerate ones, which return no values at all or always the same value). In linguistic terms, every word whose sense is an intensional construction has a denotation, but will lack a reference at some or all $\langle w, t\rangle$ pairs, in case its denotation (a partial function) fails to return a value. This applies to, *inter alia*, 'The pope', 'The first man to run 100 m under 9 s', 'The Evening Star', or 'John's wife'.

Definition 4 (*free and bound variables*). Let C be a construction with at least one occurrence of a variable ξ.

i) Let C be ξ. Then the *occurrence of ξ in C* is *free*.
ii) Let C be ${}^0X$. Then every *occurrence of ξ in C* is *^{0}bound* ('Trivialization-bound').
iii) Let C be $[\lambda x_1...x_n\ Y]$. Any *occurrence of ξ* in Y that is one of x_i, $1 \le i \le n$, is *λ-bound* in C unless it is *^{0}bound in Y*. Any *occurrence of ξ in Y* that is neither *^{0}bound* nor *λ-bound* in Y is *free* in C.
iv) Let C be $[X\ X_1...X_n]$. Any *occurrence of ξ* that is *free*, *^{0}bound*, *λ-bound* in one of $X, X_1,...,X_n$ is, respectively, *free, ^{0}bound, λ-bound in C.*
v) Let C be ${}^1X$. Then any *occurrence of ξ* that is *free, ^{0}bound, λ-bound* in X is, respectively, *free, ^{0}bound, λ-bound in C.*
vi) Let C be ${}^2X$. Then any *occurrence of ξ* that is *free, λ-bound* in a constituent of C is, respectively, *free, λ-bound in C.* If an occurrence of ξ is *^{0}bound* in a constituent ${}^0D$ of C and this occurrence of D is a constituent of X' v-constructed by X, then if the occurrence of ξ is free, λ-bound in D it is *free, λ-bound in C.* Otherwise, any other occurrence of ξ in C is *^{0}bound* in C.
vii) An *occurrence* of ξ is *free*, *λ-bound*, *^{0}bound* in C only due to (i)-(vi).

A construction with at least one occurrence of a free variable is an *open construction.* A construction without any free variables is a *closed construction.*

Definition 5 (*v-congruent and equivalent constructions*) Let C, $D/*_n \rightarrow \alpha$ be constructions, and $\approx_v /(o*_n*_n)$, $\approx /(o*_n*_n)$ binary relations between constructions of order n. Using infix notation without Trivialization, $C \approx_v D$, $C \approx D$, we define: C, D are *v-congruent*, $C \approx_v D$, iff either C and D v-construct the same α-entity, or both C and D are v-improper; C, D are *equivalent*, $C \approx D$, iff C, D are v-congruent for all valuations v. □

Corollaries. If C, D are *identical*, then C, D are equivalent, but not *vice versa.* If C, D are equivalent, then C, D are v-congruent, but not *vice versa.*

If meanings of expressions E_1, E_2, that is the constructions expressed by them, are merely v-congruent, we will say that E_1, E_2 are *co-referential.* If meanings of expressions E_1, E_2 are equivalent, we will say that E_1, E_2 are *co-denotational* or *equivalent.*

The next notion we need to define is that of *synonymy*. Our notion of synonymy is defined in terms of *procedural isomorphism.* The term 'procedural isomorphism' is a nod to Carnap's *intensional isomorphism* and Church's *synonymous isomorphism.* Church's Alternatives (0) and (1) leave room for additional Alternatives.[8] One would be Alternative (½), another Alternative (¾). The former includes α- and η-conversion while the latter adds a restricted β-conversion. If we must choose, we would prefer Alternative (¾) to soak up those differences between β-transformations that concern only λ-bound variables and thus (at least appear to) lack natural-language counterparts.

One reason for excluding unrestricted β-conversion is the well-known fact that β-conversion is not an equivalent transformation in logics boasting *partial functions*, such as TIL. Another reason is that occasionally even β-equivalent constructions have different natural-language counterparts; witness the difference between attitude reports *de dicto* vs. *de re*. Thus, the difference between "*a* believes that *b* is happy" and "*b* is believed by *a* to be happy" is just the difference between β-equivalent meanings. Where attitudes are construed as in possible-world semantics, as relations to intensions (rather than hyperintensions), the former (*de dicto*) receives the analysis

$$\lambda w \lambda t\, [{}^0Believe_{wt}\ {}^0a\ \lambda w \lambda t\, [{}^0Happy_{wt}\ {}^0b]]$$

while the latter (*de re*) receives the analysis

$$\lambda w \lambda t\, [\lambda x\, [{}^0Believe_{wt}\ {}^0a\ \lambda w \lambda t\, [{}^0Happy_{wt}\, x]]\ {}^0b]$$

Types: $Happy/(o\iota)_{\tau\omega}$; $x \rightarrow_v \iota$; a, b/ι; $Believe/(o\iota o_{\tau\omega})_{\tau\omega}$.

The *de dicto* variant is the β-equivalent contractum of the *de re* variant. Both variants are equivalent because they construct one and the same proposition, the two sentences denoting the same truth-condition. Yet they denote this proposition in *different ways*, thus they are not synonymous. The equivalent β-reduction leads here to a *loss of analytic* information, namely loss of information about *which* of the two ways, or constructions, has been used to construct this proposition.[9] In this case the loss seems to be harmless, though, because there is only one, unambiguous way to β-expand the *de dicto* version into its equivalent *de re* variant.[10]

[8] For Church's Alternatives see Anderson (1998).

[9] For the notion of analytic information, see Duží (2010) and Duží et al (2010, §5.4).

[10] In general, *de dicto* and *de re* attitudes are not equivalent, but logically independent. Consider "*a* believes that the pope is not the pope" and "*a* believes *of* the pope that *he* is not the pope". The former, *de dicto*, variant makes *a* deeply irrational and most likely is not a true attribution, while the latter, *de re*, attribution is perfectly reasonable and most likely the right one to make. In TIL the *de dicto* variant is *not* an equivalent β-contractum of the *de re* variant due to the partiality of the office $Pope/\iota_{\tau\omega}$.

However, unrestricted equivalent β-reduction sometimes yields a loss of analytic information that cannot be restored by β-expansion. The well-known example is the sentence "John loves his wife and so does Tom". This sentence is ambiguous between two readings, sloppy and strict. On its sloppy reading John and Tom share the property of each loving their own wife (and both are exemplary husbands). On the strict reading they share the property of loving John's wife (and there are troubles on the horizon). And these are two distinct properties. Thus there are two distinct analyses of "John loves his wife":

Strict: $\lambda w \lambda t\, [\lambda x\, [{}^0Love_{wt}\ x\ [{}^0Wife\_of_{wt}\ {}^0John]\ {}^0John]$
Sloppy: $\lambda w \lambda t\, [\lambda x\, [{}^0Love_{wt}\ x\ [{}^0Wife\_of_{wt}\ x]\ {}^0John]$

But an unrestricted β-reduction turns these two redexes into one and the same contractum:

$$\lambda w \lambda t\, [{}^0Love_{wt}\ {}^0John\ [{}^0Wife\_of_{wt}\ {}^0John]]$$

A piece of analytic information has been lost and using the contractum one does not know which property should be applied to Tom.[11]

The *restricted* version of *equivalent* β-conversion we have in mind consists in substituting free variables for λ-bound variables of the same type, and will be called *β_r-conversion*. For instance, we see little reason to differentiate semantically or logically between "*b* is believed by *a* to be happy" and "*b* has the property of being believed by *a* to be happy".[12] The latter sentence expresses

$$\lambda w \lambda t\, [\lambda w' \lambda t'\ \lambda x\, [{}^0Believe_{w't'}\ a\ \lambda w \lambda t\, [{}^0Happy_{wt}\, x]]_{wt}\ b]$$

This is merely a β_r-expanded form of

$$\lambda w \lambda t\, [\lambda x\, [{}^0Believe_{wt}\, a\ \lambda w \lambda t\, [{}^0Happy_{wt}\, x]]\ b]$$

Thus we define:

Definition 5 (*procedurally isomorphic constructions: Alternative* (¾)). Let *C*, *D* be constructions. Then *C*, *D* are *α-equivalent* iff they differ at most by deploying different λ-bound variables. *C*, *D* are *η-equivalent* iff one arises from the other by η-reduction or η-expansion. *C*, *D* are *β_r-equivalent* iff one arises from the other by β_r-reduction or β_r-expansion. *C*, *D* are *procedurally isomorphic*, denoted '${}^0C \approx {}^0D$', $\approx/(o{*_n}{*_n})$, iff there are closed constructions $C_1,\ldots,C_m$, $m\geq 1$, such that ${}^0C = {}^0C_1$, ${}^0D = {}^0C_m$, and all C_i, C_{i+1} ($1 \leq i < m$) are either α-, η- or β_r-equivalent.

[11] For the solution of this problem see Duží & Jespersen (in submission).

[12] This is not to say we see no reason at all not to differentiate. For instance, it could be argued that one thing is to believe that *a* is happy and another is to believe that *a* has the property of being happy, because the latter at least appears to presuppose that the believer have the additional conceptual resources to master the notion of *property*. Thus a proper calibration of procedural isomorphism is still an open problem and it can depend on the area under scrutiny. More discussion on procedural isomorphism can be found in Jespersen (2010).

Hence we advocate for the restricted β-conversion; yet β-conversion is the fundamental rule for computing the value of the function v-constructed by $[\lambda x\, Y]$ at an argument v-constructed by a construction C. Its (unrestricted) version 'by name' is this (where $Y(C/x)$ is the result of correct substitution of a construction C for x in Y):

$$[[\lambda x\, Y]\ C] \vdash Y(C/x)$$

Due to compositionality, if C is v-improper the Composition $[[\lambda x\, Y]\ C]$ is v-improper as well. But if Y is itself a Closure then it is never v-improper.[13] Thus it may happen that the right-hand side is not equivalent to the left-hand side. For this reason we restrict the rule to C being a variable which is never v-improper.

But we do need a general rule of the λ-calculus for computing the value of a function. Fortunately, it turns out to be feasible to formulate a generally valid computational rule. A distinction familiar from programming languages based on the λ-calculus holds the key to the solution. The invalid rule above is moulded on the programming technique of calling a sub-procedure C by name: the sub-procedure itself is substituted for the 'local variable' x in the 'procedure body' Y. It is well-known among programmers that this technique can have undesirable side-effects, unlike the technique of calling a sub-procedure by value.[14] The idea is simple: execute the sub-procedure C first, and then – *provided this execution does not fail* – substitute the construction of the result ('pass by the value') for x.[15]

The substitution method comes with two special functions.[16] The polymorphous function *Sub* of type $(*_n *_n *_n *_n)$ operates on constructions as follows. When applied to constructions C_1, C_2, C_3, *Sub* returns as its value the construction D that is the result of the correct (i.e. collision-less) substitution of C_1 for C_2 in C_3. For instance, the result of the Composition $[{}^0Sub\ {}^{00}John\ {}^0x\ {}^0[{}^0Wife\_of_{wt}\ x]]$ is the Composition $[{}^0Wife\_of_{wt}\ {}^0John]$. The likewise polymorphous function *Tr* returns as its value the Trivialization of its argument. Thus the result of $[{}^0Tr\ {}^0John]$ is ${}^0John$. If the variable x ranges over ι, the Composition $[{}^0Tr\ x]$ $v(John/x)$-constructs ${}^0John$. Note one essential distinction between the function *Tr* and the construction Trivialization. Whereas the variable x is *free* in $[{}^0Tr\ x]$, the Trivialization ${}^0x$ *binds* the variable x by constructing just x independently of valuation.[17]

[13] See Definition 2, iii) and iv).

[14] A recent reference to the distinction between 'call by name' and 'call by value' is Pierce (2002, pp. 56-57). See also, for instance, Hyde (1996, Ch. 11) or Plotkin (1975).

[15] For conversion by name, see Claim 2.5 and the subsequent proof in Duží et al (2010, pp.267-268); for conversion by value, see Claim 2.6 and the subsequent proof in (*ibid.*, pp. 269-270). For the general strategy (inspired by programming languages) of distinguishing between *succeeding*, *failing*, and *aborting with error*, see also Van Eijck and Francez (1995).

[16] *Sub* is introduced in Tichý (1988, p. 75) and *Tr* at (*ibid.*, p. 68).

[17] Since TIL is a λ-calculus, all variable binding is λ-binding, except for Trivialization-binding. One area where Trivialization-binding plays a key role is *existential quantification into hyperintensional contexts*, where a quantifier is introduced with a view to binding a variable that occurs bound by Trivialization, because the variable occurs inside a Trivialized context. For discussion, see Duží et al (2010, §5.3). For improved solutions, see Duží & Jespersen (2012).

For simplicity's sake, we introduce the TIL translation of the rule of β-conversion by value in its simplified version for unary functions (generalization to *n*-ary functions is obvious):

$$[[\lambda x\ Y]\ C] \vdash {}^2[{}^0Sub\ [{}^0Tr\ C]\ {}^0x\ {}^0Y]$$

Note that the Composition on the right-hand side must undergo Double Execution. Provided C is *v*-proper, it *v*-constructs an entity, say *e*. Then the result of the first step (the substitution $[{}^0Sub\ [{}^0Tr\ C]\ {}^0x\ {}^0Y]$) is the *construction* $Y(e/x)$. The resulting construction must then be executed in order to obtain the value of the function *v*-constructed by $[\lambda x\ Y]$ at the argument *e*. Hence, *Double* Execution. Otherwise, if C is *v*-improper, the substitution fails to construct anything, because due to the compositionality constraint the whole Composition $[{}^0Sub\ [{}^0Tr\ C]\ {}^0x\ {}^0Y]$ is *v*-improper and so is ${}^2[{}^0Sub\ [{}^0Tr\ C]\ {}^0x\ {}^0Y]$ (see Definition 2, iii) and vi)). In this manner compositionality is preserved and the above rule of β-conversion by value is always valid even when C is *v*-improper.

Remark. In the project of a multi-agent system that our Laboratory of Intelligent Systems dealt with in 2004-2008 we use the computational variant of TIL, the *TIL-Script* functional programming language as the language of communication between agents.[18] In the *TIL-Script* language we apply *only* this computational rule of *conversion by value*. The reason is this. Since the construction C can be *v*-improper, we need to implement a lazy evaluation mechanism in order to evaluate C only when needed. However, the properness of C can be checked only in the run time, because valuation *v* would supply values dependently on states-of-affairs.

4 Rules for the Three Kinds of Context

At this point we have lined up everything we need in order to introduce the extensional logic of hyperintensions. Yet some may protest that extensional logic of hyperintensions sounds as an oxymoron like a roaring silence. For at least since the milestones Quine (1956) and Kaplan (1968) the validity of extensional principles, in particular of quantifying-in and existential generalization, has been fielded as a logical criterion for distinguishing

(i) extensional/ transparent/'relational' contexts from
(ii) non-extensional/opaque/'notional' contexts,

the idea being that extensional (etc.) contexts are those that validate quantifying-in.[19] And conversely, if a context resists quantifying-in, it is caught in violation of one or more of the laws of extensional logic and as such eludes full logical analysis. What we are saying is that also intensional and hyperintensional contexts may be quantified into, but that the feasibility of doing so presupposes that it be done within an *extensional logic of hyperintensional contexts*. Deploying a non-extensional logic of

[18] Project No. 1ET101940420: "*Logic and Artificial Intelligence for multi-agent systems*"; supported by the program "Information Society" of the Czech Academy of Sciences. For details see http://labis.vsb.cz/.

[19] See Forbes (1996).

hyperintensions to quantify into hyperintensional contexts would, indeed, be a non-starter, generating opacity and thereby making hyperintensional attitude contexts logically intractable. However, whether one accepts quantifying into (hyper-) intensional contexts or wants to restrict quantification to extensional contexts, like "Mary is happy", the logical question remains which contexts validate quantifying-in.

Tichý issues in (1986, p. 256; 2004, p. 654) a warning against a circular definition:

Q: When is a context extensional?

A: A context is extensional if it validates
(i) *the rule of substitution of co-referential terms* and
(ii) *the rule of existential generalization.*

Q: And when are (i), (ii) valid?

A: Those two rules are valid when applied to extensional contexts.

We steer clear of the circle by defining extensionality for
1. *hyperintensions* presenting functions,
2. *functions* (including possible-world intensions), and
3. functional values.

These three levels are squared off with three kinds of context:[20]
1. *hyperintensional contexts*, in which a *construction* is not *used* to present an object, but is itself *mentioned* as functional argument (though a construction of one order higher needs to be used to mention this lower-order construction);
2. *intensional context*s, in which a construction is *used* to present a *function* without presenting a particular value of the function (moreover, the construction does not occur within another hyperintensional context);
3. *extensional context*s, in which a construction is *used* to produce a particular *value* of the function at a given argument (moreover, the construction does not occur within another intensional or hyperintensional context).

Example. Let the types of entities be: $Periodic/(o(\tau\tau))$; $Sin/(\tau\tau)$; $Solve/(o(\iota*_1))_{\tau\omega}$: the relation-in-intension between an individual and a construction the product of which the individual is seeking; π/τ ; Tom/ι.

- *Extensional context*: functional *value* is an object of predication (functional argument):

"*sin π = 0*"

$$[[{}^0Sin\ {}^0\pi] = {}^00]$$

- *Intensional context*: function-in-extension is an object of predication:

"Sine is a periodic function"

$$[{}^0Periodic\ {}^0Sin]$$

- *Hyperintensional context*: construction ("function-in-intension") is an object of predication (a functional argument):

"Tom is solving the equation sin x = 0"

$$\lambda w\lambda t\ [{}^0Solve_{wt}\ {}^0Tom\ {}^0[[{}^0Sin\ x] = {}^00]]]$$

[20] For the definition see Duží et al (2010, § 2.6 and 2.7).

Referring again to the project on multi-agent systems, we can illustrate the need for operating on a hyperintensional level like this. Agents are “born” with a minimal ontology in order to execute primitive actions like “stop”, “move forward”, “turn left”, “turn right” in case of mobile agents, and they communicate with their fellow-agents by messaging. Thus the agents must have the ability to learn by experience; and they learn not only contingent facts (extending their knowledge base) but also new *concepts* thus gradually extending their *ontology.* Since we define concepts as closed constructions (in their normal form) agents must be able to operate on the hyperintensional level of concepts (constructions).

Imagine, for instance, that an agent *a* receives a message that *b* is looking for a car park with vacancies. However, *a* does not have in its ontology the concept of a car park with vacancies. In order that the communication can proceed smoothly, *a* may learn by asking the other agents that “A car park with vacancies is a car park some of whose parking spaces are not occupied”. The content of a query message mentioning the unknown concept and asking for such a definition (refinement of the unknown concept) is

$$[{}^0 Unrecognized\ {}^0[{}^0 Vacant\ {}^0 Car\_Park]].$$

The reply message content is then

$$[{}^0 Refine\ {}^0[{}^0 Vacant\ {}^0 Car\_Park] =$$
$${}^0[\lambda w\lambda t\ \lambda x\ [[{}^0 Car\_Park_{wt}\, x] \wedge \exists y\ [[{}^0 Space\_of_{wt}\ y\ x] \wedge \neg[{}^0 Occupied_{wt}\, y]]]]].$$

Thus $[{}^0 Vacant\ {}^0 Car\_Park]$ and the construction on right-hand side become *ex definitione* equivalent (that is constructing one and the same property) and agent *a* stores the new concept into its ontology.

A dual constraint of TIL has impact on the rules of inference. It is the constraint dictated by *properly partial functions*, which are undefined for some or all of their arguments, and *improper constructions*, which fail to produce a product. Improperness stems from the procedure of applying a properly partial function *f* to an argument *a*, such that *f* returns no value at *a.* The procedure of functional application induces an extensional context. Thus when specifying the rules of quantifying-in, we must check whether particular constituent constructions occurring extensionally are improper. If none is, the particular rule of quantifying-in is valid.

The rules of *improperness* can be schematically summarized as follows. If a Composition is used in an extensional context as a procedure of application a properly partial function $F/(\beta\alpha)$ to an argument a/α and if *F* has no-value at *a* (value gap) then

$$[{}^0F\ {}^0a] \text{ is } \textit{v-improper}$$

and so is any construction *C* occurring extensionally and containing $[{}^0F\ {}^0a]$ as a constituent; *partiality is strictly propagated up*:

$$[\ldots [\ \ldots [{}^0F\ {}^0a]\ \ldots]\ \ldots] \text{ is } \textit{improper}$$

until the context is raised up to *hyper/intensional*:

intensional:	$\lambda x \ldots [\ldots [\ \ldots [{}^0F\ {}^0a]\ \ldots]\ \ldots]$ is *proper*
hyperintensional:	${}^0[\ldots [\ \ldots [{}^0\mathrm{F}\ {}^0\mathrm{a}]\ \ldots]\ \ldots]$ is *proper*

4.1 The Rules of Existential Generalization

a) ***extensional*** context. Let $[\ldots[^0F\ ^0a]\ldots]$ v-construct the truth-value **T**. Then the following rule is truth-preserving:

$$[\ldots[^0F\ ^0a]\ldots] \mid\!\!-\!\!- \exists x\, [\ldots[^0F\, x]\ \ldots];\ x \rightarrow_v \alpha$$

Proof:

1.	$[\ldots[^0F\ ^0a]\ldots]$	assumption
2.	$[\ldots[^0F\, x]\ \ldots]$	$v(a/x)$-constructs **T**
3.	$\lambda x\, [\ldots[^0F\, x]\ldots]$	v-constructs a non-empty class
4.	$[^0\exists\lambda x\, [\ldots[^0F\, x]\ldots]]$	EG, 3

Example: $\lambda w\lambda t\, [^0Wise_{wt}\ ^0Pope_{wt}] \models \lambda w\lambda t\, \exists x\, [^0Wise_{wt}\, x]$;
Types: $Wise/(o\iota)_{\tau\omega}$; $Pope/\iota_{\tau\omega}$; $x \rightarrow_v \iota$.

b) ***intensional*** context. Let $[\ldots\lambda y\ [\ \ldots\ [^0F\ ^0a]\ \ldots]]$ v-construct **T**. Then the following rule is truth-preserving:

$$[\ldots\lambda y\ [\ \ldots\ [^0F\ ^0a]\ \ldots]] \mid\!\!-\!\!- \exists f\, [\ldots\lambda y\ [\ \ldots\ [f\ ^0a]\ \ldots]];\quad f \rightarrow_v (\beta\alpha)$$

Proof:

1.	$[\ldots\lambda y\ [\ \ldots\ [^0F\ ^0a]\ \ldots]]$	assumption
2.	$[\ldots\lambda y\ [\ \ldots\ [f\ ^0a]\ \ldots]]$	$v(F/f)$-constructs **T**
3.	$\lambda f\, [\ldots\lambda y\ [\ \ldots\ [f\ ^0a]\ \ldots]]$	v-constructs a non-empty class
4.	$[^0\exists\lambda f\, [\ldots\lambda y\ [\ \ldots\ [f\ ^0a]\ \ldots]]]$	EG, 3

Example: $\lambda w\lambda t\, [^0Believe_{wt}\ ^0b\ \lambda w\lambda t\, [^0Wise_{wt}\ ^0Pope_{wt}]] \models$
$\lambda w\lambda t\, \exists f\, [^0Believe_{wt}\ ^0b\ \lambda w\lambda t\, [^0Wise_{wt}\, f_{wt}]]$;
Additional types: $Believe/(o\iota o_{\tau\omega})_{\tau\omega}$: an intensional attitude to a proposition; $f \rightarrow_v \iota_{\tau\omega}$

Gloss: If *b* believes that the Pope is wise then there is an office such that *b* believes that its holder is wise.

c) ***hyperintensional*** context. Let $[\ldots\ ^0[\ \ldots\ [^0F\ ^0a]\ \ldots]]$ v-construct **T**. Then the following rule is truth-preserving:

$$[\ldots\ ^0[\ \ldots\ [^0F\ ^0a]\ \ldots]] \mid\!\!-\!\!- \exists c\ ^2[^0Sub\ c\ ^{00}F\ ^0[\ldots\ ^0[\ \ldots\ [^0F\ ^0a]\ \ldots]]];$$
$$c \rightarrow_v *_n;\ ^2c \rightarrow_v (\beta\alpha)$$

Proof:

1.	$[\ldots\ ^0[\ \ldots\ [^0F\ ^0a]\ \ldots]]$	assumption
2.	$^2[^0Sub\ c\ ^{00}F\ ^0[\ldots\ ^0[\ \ldots\ [^0F\ ^0a]\ \ldots]]]$	$v(^0F/c)$-constructs **T**
3.	$\lambda c\ ^2[^0Sub\ c\ ^{00}F\ ^0[\ldots\ ^0[\ \ldots\ [^0F\ ^0a]\ \ldots]]$	v-constructs a non-empty class
4.	$[^0\exists\lambda c\ ^2[^0Sub\ c\ ^{00}F\ ^0[\ldots\ ^0[\ \ldots\ [^0F\ ^0a]\ \ldots]]]$	EG, 3

The step 2 may be difficult to understand. Here is an additional explanation. The Composition $[^0Sub\ c\ ^{00}F\ ^0[\ldots\ ^0[\ \ldots\ [^0F\ ^0a]\ \ldots]]]$ $v(^0F/c)$-constructs the Composition $[\ldots\ ^0[\ \ldots\ [^0F\ ^0a]\ \ldots]]$. In order to v-construct **T**, this Composition must be executed again. Therefore, Double Execution.

Example: $\lambda w \lambda t\ [^0 Believe^*_{wt}\ ^0 b\ ^0[\lambda w \lambda t\ [^0 Wise_{wt}\ ^0 Pope_{wt}]]\ \models$
$\lambda w \lambda t\ \exists c\ [^0 Believe^*_{wt}\ ^0 b\ [^0 Sub\ c\ ^{00} Pope\ ^0[\lambda w \lambda t\ [^0 Wise_{wt}\ ^0 Pope_{wt}]]]]$;
Additional types: $Believe^*/(o\iota{*_n})_{\tau\omega}$: a hyperpropositional belief; $c \rightarrow *_n$; $^2c \rightarrow_v \iota_{\tau\omega}$

Gloss: If b (explicitly) believes* that the Pope is wise, then there is a concept of an office such that b believes* that the holder of the office is wise.

Note: In this example the $[^0 Sub\ c\ ^{00} Pope\ ^0[\lambda w \lambda t\ [^0 Wise_{wt}\ ^0 Pope_{wt}]]]$ is not Double executed, because b is related just to the Composition itself constructed by this substitution.

Hyperpropositional attitudes must be used if the attributer is reproducing faithfully *b*'s perspective. For instance, suppose that the office of Pope is identical with the office of the Bishop of Rome. Then it may be the case that though *b* believes that the Pope is wise, he may disbelieve that the Bishop of Rome is wise.

4.2 Leibniz's Rule of Substitution in the Three Kinds of Context

a) In an ***extensional context*** substitution *salva veritate* of *v-congruent* constructions is valid.
Example.

"The president of CR is the husband of Livia Klausova"
"The president of CR is an economist"

"The husband of Livia Klausova is an economist"

Proof:
1) $\lambda w \lambda t\ [^0 President\_of_{wt}\ ^0 CR]_{wt} \approx_v \lambda w \lambda t\ [^0 Husband\_of_{wt}\ ^0 Livia]_{wt}$ assumption
2) $[^0 Economist_{wt}\ \lambda w \lambda t\ [^0 President\_of_{wt}\ ^0 CR]_{wt}]$ assumption

---

3) $[^0 Economist_{wt}\ \lambda w \lambda t\ [^0 Husband\_of_{wt}\ ^0 Livia]_{wt}]$ Leibniz, 2)

Types. $President\_of/(\iota\iota)_{\tau\omega}$; CR/ι; $Husband\_of/(\iota\iota)_{\tau\omega}$; $Livia/\iota$; $Economist/(o\iota)_{\tau\omega}$;

b) In an ***intensional context*** substitution *salva veritate* of *equivalent* constructions is valid.
Example.

"The president of CR is the highest representative of CR"
"Tom wants to become the president of CR"

"Tom wants to become the highest representative of CR''

Proof:
1) $\lambda w \lambda t\ [^0 President\_of_{wt}\ ^0 CR] \approx \lambda w \lambda t\ [^0 Highest\_Rep\_of_{wt}\ ^0 CR]$ assumption
2) $[^0 Want\_be_{wt}\ ^0 Tom\ \lambda w \lambda t\ [^0 President\_of_{wt}\ ^0 CR]]$ assumption

---

3) $[^0 Want\_be_{wt}\ ^0 Tom\ \lambda w \lambda t\ [^0 Highest\_Rep\_of_{wt}\ ^0 CR]]$ Leibniz, 2)

Additional types. *Highest_Rep_of*/$(\iota\iota)_{\tau\omega}$; *Want_be*/$(o\iota\iota_{\tau\omega})_{\tau\omega}$: the relation-in-intension of an individual to an individual office; *Tom*/ι.

c) In a ***hyperintensional context*** substitution *salva veritate* of *procedurally isomorphic* constructions is valid.

Example. Suppose that 'azure' and 'sky-blue' are synonymous.

$$\frac{\text{“Tom believes* that Marie's blouse is azure”}}{\text{“Tom believes* that Marie's blouse is sky-blue”}}$$

Proof:

1) $[{}^0Believe^*_{wt}\ {}^0Tom\ {}^0[\lambda w\lambda t\ [{}^0Azure_{wt}\ [{}^0Blouse\_of_{wt}\ {}^0Marie]]]]$ assumption

2) ${}^{00}Azure \approx {}^{00}Sky\_Blue$ assumption

3) $[{}^0Believe^*_{wt}\ {}^0Tom\ {}^0[\lambda w\lambda t\ [{}^0Azure_{wt}\ [{}^0Blouse\_of_{wt}\ {}^0Marie]]]]$ Leibniz

Some might object that this argument is invalid, because it is possible that Tom believes that Marie's blouse is azure without believing that Marie's blouse is sky-blue. We disagree and on this point we refer to Richard who says:

> "… It is impossible for a (normal, rational) person to understand expressions which have identical senses but not be aware that they have identical senses." (2001, pp. 546-7)

Hence the paradox of analysis is *not* a problem of hyperintensionality. Rather, it is a matter of linguistic incompetence (failure to recognize pairs of synonyms) and *not of logical incompetence* (failure to recognize pairs of procedurally isomorphic hyperintensions).

5 Conclusion

Quantifying into hyperintensional contexts requires an *extensional logic of hyperintensions*. Much non-trivial footwork is required to lay out such a large-scale logical semantics. Once this is done, though, quantifying into hyperintensional and intensional contexts turns out to be as trivially valid as quantifying into extensional contexts. However quantifying into hyperintensional contexts introduces a *technical complication* absent in quantifying into intensional and extensional contexts. In a hyperintensional context a construction occurs *mentioned* (as an argument of another function) rather than used (to construct a function). The complication is that, since every constituent of a *mentioned* construction itself occurs mentioned, the quantifier cannot bind any variables inside the hyperintensional context, thus rendering quantifying-in impossible. The solution consists in applying a *substitution technique* that makes the variables amenable to binding. Moreover, based on the substitution method we introduced the generally valid computational rule of a partial lambda calculus, *viz.* reduction 'by value'.

Once the three kinds of context, namely extensional, intensional and hyperintensional are defined, the substitution of identicals is trivially valid. There is no cogent reason for invalidity of Leibniz's law. Only that we must substitute that object which is the object of predication in a given context.

Acknowledgements. This research has been supported by the Grant Agency of the Czech Republic, project No. 401/10/0792, "Temporal aspects of knowledge and information" and also by the internal grant agency of VSB-TU Ostrava, project No. SP/2010214 "Modelling, simulation and verification of software processes II".

References

Anderson, C.A.: Alonzo Church's contributions to philosophy and intensional logic. The Bulletin of Symbolic Logic 4, 129–171 (1998)

Bealer, G.: Quality and Concept. Clarendon Press, Oxford (1982)

Carnap, R.: Meaning and Necessity. Chicago University Press, Chicago (1947)

Church, A.: Intensional isomorphism and identity of belief. Philosophical Studies 5, 65–73 (1954)

Church, A.: A revised formulation of the logic of sense and denotation. Alternative (1). Noûs 27, 141–157 (1993)

Cleland, C.E.: On effective procedures. Minds and Machines 12, 159–179 (2002)

Cresswell, M.J.: Hyperintensional logic. Studia Logica 34, 25–38 (1975)

Cresswell, M.J.: Structured meanings. MIT Press, Cambridge (1985)

Duží, M.: The paradox of inference and the non-triviality of analytic information. Journal of Philosophical Logic 39(5), 473–510 (2010)

Duží, M., Jespersen, B., Materna, P.: Procedural Semantics for Hyperintensional Logic. Foundations and Applications of Trasnsparent Intensional Logic, 1st edn. Logic, Epistemology, and the Unity of Science, vol. 17. Springer, Berlin (2010)

Duží, M., Jespersen, B.: Transparent quantification into hyperintensional contexts de re. Logique and Analyse 220 (December 2012) (to appear)

Duží, M., Materna, P.: Can concepts be defined in terms of sets? Logic and Logical Philosophy 19, 195–242 (2010)

van Eijck, J., Francez, N.: Verb-phrase ellipsis in dynamic semantics. In: Masuch, M., Polos, L. (eds.) Applied Logic: How, What and Why?, pp. 29–60. Kluwer (1995)

Forbes, G.: Substitutivity and the coherence of quantifying in. The Philosophical Review 105, 337–371 (1996)

Fox, C., Lappin, S.: A framework for the hyperintensional semantics of natural language with two implementations. Lecture Notes in Computational Linguistics 2009, 175–192 (2001)

Frege, G.: Über Sinn und Bedeutung. Zeitschrift für Philosophie und philosophische Kritik 100, 25–50 (1892)

Hyde, R.: The Art of Assembly Language Programming (1996), `http://www.arl.wustl.edu/~lockwood/class/cs306/books/artofasm/toc.html` (retrievable)

Jespersen, B.: Why the tuple theory of structured propositions isn't a theory of structured propositions. Philosophia 31, 171–183 (2003)

Jespersen, B.: How hyper are hyperpropositions? Language and Linguistics Compass 4, 96–106 (2010)

Kaplan, D.: Quantifying in. Synthese 19, 178–214 (1968)

Kaplan, D.: Opacity. In: Hahn, L. (ed.) The Philosophy of W.V. Quine, pp. 229–289. Open Court, La Salle (1986)

Kaplan, D.: Dthat. In: Cole, P. (ed.) Syntax and Semantics, vol. 9, Academic Press, New York (1990); reprinted in: Yourgrau (ed.) Demonstratives. Oxford University Press, Oxford

Klement, K.C.: Frege and the Logic of Sense and Reference. Routledge, New York (2002)

Kripke, S.: Semantical considerations on modal logic. Acta Pilosophica Fennica 16, 83–94 (1963)

Lewis, C.I.: A Survey of Symbolic Logic. University of California Press, Berkeley (1918)

Lewis, D.: General semantics. In: Davidson, D., Harman, G. (eds.) Semantics of Natural Language, pp. 169–218. Reidel, Dordrecht (1972)

Moschovakis, Y.N.: Sense and denotation as algorithm and value. In: Väänänen, J., Oikkonen, J. (eds.) Lecture Notes in Logic, vol. 2, pp. 210–249. Springer, Berlin (1994)

Moschovakis, Y.N.: A logical calculus of meaning and synonymy. Linguistics and Philosophy 29, 27–89 (2006)

Pierce, C.B.: Types and Programming Languages. MIT Press, London (2002)

Plotkin, G.D.: Call-by-name, call-by-value and the λ-calculus. Theoretical Computer Science 1, 125–159 (1975)

Quine, W.v.O.: Quantifiers and propositional attitudes. Journal of Philosophy 53, 177–186 (1956)

Richard, M.: Analysis, synonymy and sense. In: Anderson, C.A., Zeleny, M. (eds.) Logic, Meaning and Computation: Essays in Memory of Alonzo Church. Synthese Library, vol. 305, pp. 545–571. Kluwer, Dordrecht (2001)

Tichý, P.: Smysl a procedura. Filosofický časopis 16, 222–232 (1968); translated as 'Sense and procedure' in Tichý, 77–92 (2004)

Tichý, P.: Intensions in terms of Turing machines. Studia Logica 26, 7–25 (1969); reprinted in Tichý, 93–109 (2004)

Tichý, P.: The Foundations of Frege's Logic. De Gruyter, Berlin (1988)

Tichý, P.: Collected Papers in Logic and Philosophy. In: Svoboda, V., Jespersen, B., Cheyne, C. (eds.), Filosofia, Czech Academy of Sciences, and Dunedin: University of Otago Press, Prague (2004)

Culture Sensitive Aspects in Software Engineering

Hannu Jaakkola

Tampere University of Technology, Pori
P.O. Box 300, FI-28101 Pori, Finland
hannu.jaakkola@tut.fi

Abstract. The characteristics of software engineering (SE) are changing rapidly. The following trends are easy to notice: the transfer from plan driven development to agile development, the transfer towards distributed and multicultural teams and organization structure, the increasing importance of services related to software products or software itself, transfer towards cloud implementation of information systems. Even as agile software development is encouraging active interaction inside teams and between the developers and the clients, distributed work is increasing its difficulty. The problems of distribution itself can be solved by tools and techniques, e.g. by improved version and configuration management, careful asset repository management, tools forcing the production of unified specifications, and tools supporting communication in a distributed development context. When software organizations are multicultural, one additional dimension of difficulty appears. Even in a single unit, differences in cultural background may cause problems, but the problems become emphasized especially in the case of distributed work. The same problem also appears in software related services: to an increasing extent the service chain is distributed across cultural borders. Process models are used to provide means for the better management of software engineering and services. Predefined processes force the developers to follow the given guidelines throughout the organization – regardless of the geographical location and cultural background of the employees. This is also the expectation of managers. A slightly more careful look at the real situation gives a different view: some processes are more culture sensitive than others, and the practices are "tuned" to follow the rules of the culture. This paper opens up the discussion on the cultural aspects in connection with software engineering, taking into account especially the role of national cultures.

Keywords: Software Engineering, Cross-Cultural, Multicultural, Software Life Cycle Processes, Culture Sensitivity, Globalization, Globalized Software Development, Cultural Analysis.

1 Introduction

In our global and open world even the ICT (Information and Communication Technology) industry is becoming more global. The distribution of work has already been a part of daily activities in the software industry for a long time – mainly driven by the networking of collaborative software companies and also as a consequence of company acquisitions and mergers. Software engineering is team-oriented work, in which the role of successful communication is essential. In software engineering two basic forms of communication can be recognized: interaction between people and documentation. Interaction between people is essential in teams responsible for one development phase and documentation is a means to transfer knowledge between development phases. This is actually the situation only in traditional software engineering made by local teams in one development site. In distributed work even a part of human interaction is replaced by rules and documentation.

In distributed work effective communication is more challenging. Two types of problems appear: how to support team-level communication, and how to manage the assets developed. The latter problem is usually solved by tools related to version, configuration, and asset management. The first problem is more difficult to solve. Paasivaara et al. (2010, p. 3-29) have reported the results of their study concerning communication in distributed (global) software companies. The following reasons are seen as the source of problems:

- lack of informal communication,
- misunderstandings in communication,
- cultural problems (both at national and organizational level),
- limited traveling (to avoid costs),
- lack of trust between the remote sites,
- time given to problem solving is too short,
- differences in processes and working practices between the remote sites,
- differences in tools and communication technology,
- time-zone differences.

Some of the problems listed above are related to the distribution of work, but some also to cultural differences in different parts of the organization or also inside one organizational unit. Although it is important to understand the communication problems related to the distributed characteristics of the work, in this paper we will concentrate on the cultural aspects only. Paasivaara et al. (ibid.) also give a set of guidelines and good practices for avoiding problems:

- the use of frequent deliveries,
- reduction of temporal distance,
- reduction of the number of sites,
- reduction of the cultural distance,
- the use of dedicated team members and creation of single teams,
- need to organize face-to-face communication opportunities,
- agreeing (tested) communication practices and tools,
- organizing progress monitoring and visibility across sites.

Special emphasis is given to the appropriate use of communication tools.

A typical way to manage the complexity related to software engineering is the use of the process-oriented approach to recognize and specify the different activities needed in software development. The principles derived from the process models are implemented in the quality management system of the organizations. One commonly used process model is based on the work of the International Standardization Organization (ISO). The standard ISO/IEC 12207 (2008) separates system and software specific process areas and divides them into seven process groups. The processes relevant to a company are specified and further - in the case of Quality Management System (QMS) certification - assessed according to the selected assessment standard / guide. One of the standards developed for this purpose is ISO 15504 (2006). The purpose of the assessment is to prove that the capability level requirements specified by the standard are reached. The maturity of an organization is based on the capability profile of the processes.

The basic idea behind the process-oriented approach is that all parts of the organization will follow the same rules and implement the operations in the same way. In global software organizations the situation, however, is not so simple. Jaakkola et al. (2011) have reported the results of interviews with some software companies that have globalized their operations. The companies reported the similarity of their processes over the organization. Closer scrutiny of their activities, however, shows that to some extent the processes have been adapted to take into account the differences in the cultural background of the employees. The human related (leadership) and organizational activities are typically adapted to take into account the local circumstances and pure engineering related work follows the non-adapted flow of operations. In addition, the role of documentation increases when the opportunity for human-to-human communication is either missing or done over a network. To simplify the former: the more demanding the implementation of (face-to-face) communication, the more focus must be given to the exact specification in the form of documentation. Additionally, the role of architectures is rising in importance – a predefined architecture and architectural style supports independence between development teams. The role of architectures in software engineering is discussed e.g. in (Jaakkola and Thalheim 2011; 2010; 2005).

This paper reports the findings of the STEP project (Steps in Multi-Cultural Software Business Globalization; 2009-2011). The results are published, in addition to several journal and conference papers, in the form of three M.Sc. theses. Karttunen (2010) has developed an assessment model to be used in analyzing the culture sensitivity of service processes recognized by the ITIL framework. ITIL - The Information Technology Infrastructure Library - is a set of concepts and practices for Information Technology Services Management (see. e.g. itSMF International, 2011). The thesis by Wesslin (2010) introduces a model to be used in competence transfer between global sites of an organization. The thesis by Statkaityte (2011) analyzes the culture sensitivity of software processes. In addition, the thesis gives an overview of differences between selected cultures, especially from the software engineering point of view.

This paper opens up the discussion on the role of national cultures in software engineering work. It is based on the findings of the STEP project and introduces a model to be used in the analysis of the culture sensitivity of software engineering related processes. Section 2 provides a short introduction to the process structure related to software engineering. Section 3 introduces commonly used frameworks used in analyzing cultural differences. Section 4 combines the two former approaches first in the form of related studies and then by integrating cultural aspects in software process models. Section 5 introduces a tool for analyzing the culture sensitivity of processes. Section 6 concludes the paper.

2 Software Engineering Process Framework

2.1 Maturity of Organizations and Capability of Processes

The first formal description developed to manage the complexity of software engineering was published by William Royce (1970). His waterfall life cycle model has been the root of several variations and applications (see e.g. Pfleeger and Atlee, 2006). The basic idea to divide the life cycle of software into consecutive steps made development work visible and provided milestones to follow the realization of the development schedule. Although the waterfall model was a simple formalization of life cycle processes and did not recognize the role of supporting activities, it has also played a significant role in the development of the process models currently used, which in addition to pure engineering processes, also identify and classify a set of related activities. The Capability and Maturity Model combines process structure with the measurement framework used to analyze the capability of (relevant) processes in the organization.

The basic concept of the Capability and Maturity Model (CMM) was introduced first by Philip Crosby (1979) as a part of his quality management framework. The principle was applied in software development by Watts Humphrey (1989). His Capability Maturity Model for Software has been further improved and developed, first by applying it in different sectors of software related activities, and later by integrating the separate models in CMMI (Capability Maturity Model Integration). The current version is CMMI V 1.3 (SEI 2011).

Simultaneously to CMMI work, the International Standardization Organization (ISO) started a project to develop an international standard for process improvement in the software industry. The Software Process Improvement and Capability Determination project (SPICE) was organized as a part of ISO/IEC JTC1/SC7 activities and the result was published as a series of ISO 15504-x standards; see e.g. (ISO/IEC, 2005). Nowadays CMMI and ISO 15504 are more or less unified. The original process-oriented approach of SPICE was also adopted in CMMI in the form of a *continuous* (process-oriented) model to complete the original organization-oriented *staged model.*

The concepts "maturity" and "capability" indicate sophistication and good quality. In the context of the model *maturity* is connected to the organization and *capability* to the processes. The terms are also connected to each other: a precondition for a high maturity level organization is that its (relevant) processes also possess high capability. Both the continuous and staged approaches implement the idea of continuous improvement in an organization – either targeted in the selected processes or in the organization. Although we will not go into detail concerning capability / maturity analysis itself, a short overview of the topic is worth giving. The capacities are characterized by a five (six)-step scale (simplified definitions):

1. *Not* executed (only in the continuous model)
2. *Initial* (Executed): Processes are unpredictable, poorly controlled, and reactive. They are (typically) undocumented and in a state of dynamic change, tending to be driven in an ad hoc, uncontrolled and reactive manner by users or events. This provides a chaotic or unstable environment for the processes.
3. *Managed* (Repeatable): The process is characteristic of projects and is usually reactive. Processes at this level are repeatable, possibly with consistent results.
4. *Defined*: The process is characteristic of an organization and is usually proactive. Sets of defined and documented standard processes are established and subject to some degree of improvement over time.
5. *Quantitatively managed*: The process is measured and controlled using process metrics and management.
6. *Optimizing*: The focus is on process improvement - continually improving process performance through both incremental and innovative changes.

The quality improvement path (from a lower to a higher level) is specified by the model. Where lower levels indicate flexibility and variance, the higher levels indicate formal behavior and stiffness. In relation to the topic of this paper – the flexibility of processes to take cultural differences into account – the approach is opposite. However, our aim is to find a solution that on the one hand fills the needs of capability / maturity assessment and on the other hand also allows culture awareness in high capability processes and mature organizations.

2.2 Software Life Cycle Processes

The process model groups the activities that may be performed during the life cycle of a software system. The process life cycle model (Fig. 1) applied in this paper is adapted from the ISO/IEC 15504 (2005) process assessment standard (adapted from ISO/IEC 12207 – 2004). The CMMI process model was rejected as an alternative because of its conciseness and poorer suitability for process-oriented analysis. The newer ISO/IEC process model (ISO/IEC 12207: 2008) was rejected because the assessment standard is not yet harmonized and is still based on the old version of ISO/IEC 12207. *Primary Life Cycle Processes* consist of processes that serve primary interest groups during the life cycle of the software. A primary interest group is one that initiates or performs the development, operation, or maintenance of software products, e.g. the acquirer, the supplier, the developer, the operator, and the maintainer. *Supporting Life Cycle processes* support another process as an integral part, each with a distinct purpose. A supporting process contributes to the success and quality of the software project. A supporting process is employed and executed, as needed, by another process. *Organizational Life Cycle Processes* are employed by an organization to establish and implement an underlying structure needed by associated life cycle processes. The organizational processes are not usually directly connected to specific projects and contracts, but good project practices derived from projects are adopted to improve the organization.

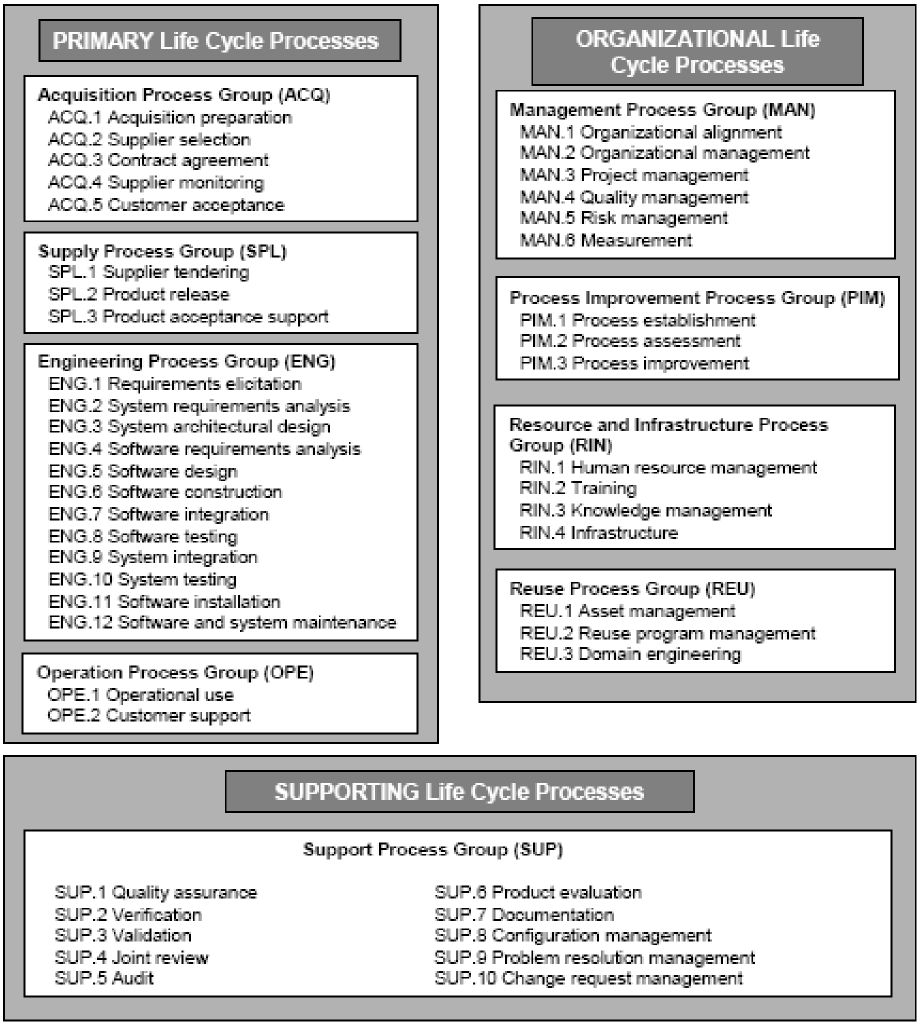

PRIMARY Life Cycle Processes
Acquisition Process Group (ACQ)
ACQ.1 Acquisition preparation
ACQ.2 Supplier selection
ACQ.3 Contract agreement
ACQ.4 Supplier monitoring
ACQ.5 Customer acceptance
Supply Process Group (SPL)
SPL.1 Supplier tendering
SPL.2 Product release
SPL.3 Product acceptance support
Engineering Process Group (ENG)
ENG.1 Requirements elicitation
ENG.2 System requirements analysis
ENG.3 System architectural design
ENG.4 Software requirements analysis
ENG.5 Software design
ENG.6 Software construction
ENG.7 Software integration
ENG.8 Software testing
ENG.9 System integration
ENG.10 System testing
ENG.11 Software installation
ENG.12 Software and system maintenance
Operation Process Group (OPE)
OPE.1 Operational use
OPE.2 Customer support
ORGANIZATIONAL Life Cycle Processes
Management Process Group (MAN)
MAN.1 Organizational alignment
MAN.2 Organizational management
MAN.3 Project management
MAN.4 Quality management
MAN.5 Risk management
MAN.6 Measurement
Process Improvement Process Group (PIM)
PIM.1 Process establishment
PIM.2 Process assessment
PIM.3 Process improvement
Resource and Infrastructure Process Group (RIN)
RIN.1 Human resource management
RIN.2 Training
RIN.3 Knowledge management
RIN.4 Infrastructure
Reuse Process Group (REU)
REU.1 Asset management
REU.2 Reuse program management
REU.3 Domain engineering
SUPPORTING Life Cycle Processes
Support Process Group (SUP)
SUP.1 Quality assurance
SUP.2 Verification
SUP.3 Validation
SUP.4 Joint review
SUP.5 Audit
SUP.6 Product evaluation
SUP.7 Documentation
SUP.8 Configuration management
SUP.9 Problem resolution management
SUP.10 Change request management

Fig. 1. Life Cycle Processes (ISO 15504-5:2005)

Each process in ISO 12207 (2004) is described in terms of the following attributes:

- The title describes the scope of the process;
- The purpose describes the goals of performing the process;
- The outcomes express the observable results;
- The activities (base practices) are a list of actions that are used to achieve the outcomes;
- The tasks are requirements, recommendations, or permissible actions intended to support the achievement of the outcomes.

In addition, an essential part of the process description comprises the work products, which have either the role of input or output (Fig. 2).

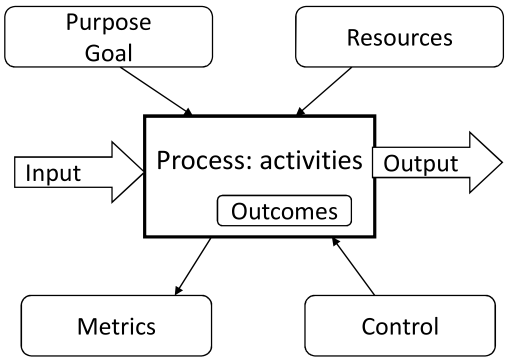


Fig. 2. The key elements of a process

Process metrics are used for two purposes: (1) to measure the properties of the process to allow control as feedback to guide further steps and activities in the process; (2) to give information about the capability of the process for process assessment.

3 Frameworks for Cultural Analysis

The role of cultures in different contexts has been the topic of several studies. Two quite widely used frameworks have been adopted for wide use. The framework of *Geert Hofstede* consists of six cultural dimensions differing between cultures (Hofstede 2005; 2010; 2011). The (relative) score value of each dimension considers relations between people, definition of the self and others, acceptability of selected management practices and organizational structure, values, orientation towards risks, and time span in relation to work. The *Lewis Model* of Culture (Lewis 1999; Lewis 2011) focuses on communication and the interaction skills of people. The basic profiles are linear active, multi-active, and reactive. National cultures are located in the continuums of a triangle with the basic profiles in the corners. The cultures differ from each other in values and core beliefs, concept of time, communication patterns, listening habits, leadership style, self-image, motivation factors, manners and taboos, and organizing their work.

Hofstede's and Lewis' models can be seen as a layered whole. The former represents national culture based basic behavior and the latter the behavioral issues of an individual. The models provide a macro-level framework for studying cultural factors. However, in modeling, designing, and implementing cross-cultural knowledge in software engineering, a more detailed contextual analysis is needed i.e. application, situational, task- and user-specific analysis. A short overview of the models is given below.

Hofstede's framework of national cultures

Hofstede's (Hofstede and Hofstede 2005; Hofstede *et al.*, 2010) framework consists of six dimensions having relative score values. Four of them belong to the original model:

- *Individualism / Collectivism (IDV)* describes the extent to which a society emphasizes the individual or the group. Individualistic societies encourage their members to be independent and look out for themselves; collectivistic societies emphasize the group's responsibility for each individual.
- *Power distance (PDI)* describes the extent to which a society accepts that power is distributed unequally. In high PDI societies individuals prefer little consultation between superiors and subordinates and in low PDI societies individuals prefer consultative styles of leadership.
- *Masculinity/Femininity (MAS)* refers to the values more likely to be held in a society. Masculine societies are characterized by an emphasis on money and things. Feminine cultures are characterized by concerns for relationships, nurturing, and quality of life.
- *Uncertainty avoidance (UAI)* refers to the extent that individuals in a culture are comfortable (or uncomfortable) with unstructured situations. High UAI societies prefer stability, structure, and precise managerial direction, and in low UAI societies people accept ambiguity, unstructured situations, and broad managerial guidance to a greater extent.

The values of these four indices are available for all countries investigated by Hofstede. In later studies two new indices were added – LTO first, and WVS in the latest book of Hofstede et al. (2010).

- *Long-term/Short-term orientation (LTO)* refers to the extent to which a culture programs its members to accept delayed gratification of their material, social, and emotional needs. Business people in long-term oriented cultures are accustomed to working toward building strong positions in their markets and do not expect immediate results. In short-term oriented cultures, the "bottom line" (the results of the past month, quarter, or year) is a major concern. Control systems focus on it and managers are constantly judged by it.
- *Well-being versus Survival (WVS)* indicates the acceptance of indulgence connected to enjoyable life and happiness. Restrained cultures emphasize abstaining from indulgence and reserved behavior, whereas indulgent cultures are permissive. A high WVS score is associated with the combination of high IDV and low MAS.

Although all dimensions are generalizations and individuals may vary from their society's descriptors, the values are worth taking into account when organizing work in a distributed multicultural context. A very generalized conclusion of the studies of the author and his research group (Jaakkola 2009; Jaakkola and Heimbürger 2009; Jaakkola et al. 2010; 2011) indicate that multicultural teams with a large difference in index values are challenging to manage and that special emphasis must be given to leadership practices and organizing the work.

Lewis' model of culture

The Lewis Model of Culture focuses on communication and interaction skills (Lewis 1999; 2011). Cultural behavior is not accidental, but the end product of millennia of collected wisdom, filtered and passed down through hundreds of generations and translated into hardened, core beliefs, values, notions, and persistent action patterns. Fig. 3 points out the key structure behind the model.

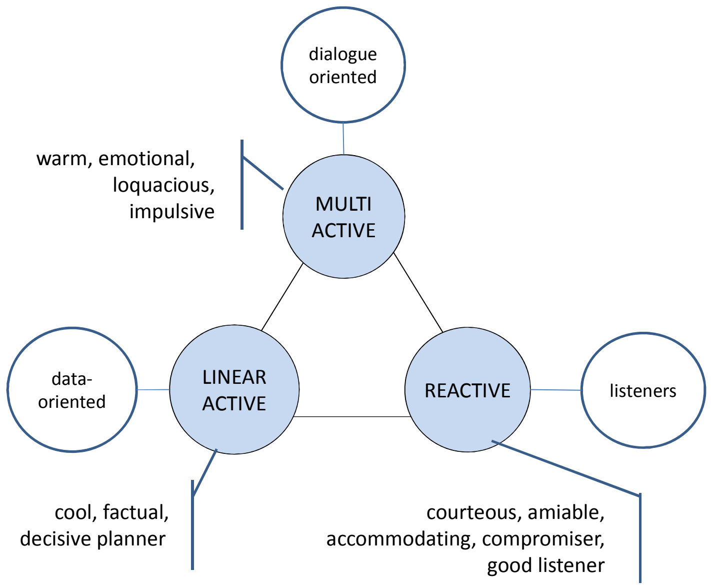

dialogue oriented
warm, emotional, loquacious, impulsive
MULTI ACTIVE
data-oriented
LINEAR ACTIVE
REACTIVE
listeners
cool, factual, decisive planner
courteous, amiable, accommodating, compromiser, good listener

Fig. 3. Lewis' model of culture (original Lewis (2000); modified by Statkaityte, (2011) and the author)

Cultures are classified in three distinctive categories: *multi-active, linear-active, and reactive*. The corresponding classification in information gathering are *data-oriented, dialogue-oriented,* and *listeners*. Data-oriented and dialogue-oriented are more explicit terms for low-context and high-context cultures, and a third group, ”listening culture” has been added to describe reactive Asians, who embrace information technology but are also effective networkers.

People from a *multi-active culture* are typically people-oriented, talkative, and emphasize interpersonal relationships (Slavs, Africans, Latinos). People of this profile talk most of the time and do several things at once, plans are outlines, they are emotional, polite and display their feelings, they interrupt often and put feelings before facts;, they are people-oriented, their truth is flexible; and they always have good excuses/explanations?.

People from *reactive cultures* are typically introverted, respect-oriented, and emphasize listening (Chinese, Koreans, Vietnamese). People in reactive cultures are listeners and able to react to their partner's actions, they look at general principles, they are indirect and conceal feelings, losing face is not acceptable and they never confront, they are diplomatic (over truth), very people-oriented and never interrupt, statements are promises for them.

People from *linear-active cultures* are typically task-oriented, highly organized, and emphasize planning (Scandinavians, Australians, Americans). They talk as needed, do one thing at a time, plan their activities step by step, they are polite but direct and confront with logic, they dislike losing face, they are job-oriented and prefer truth before diplomacy.

Cultural analysis of some countries

In our project a set of six globally operating software companies were investigated to obtain knowledge and understanding of their globalization practices. The results are briefly introduced in Section 4 of this paper. In this context we make a cultural analysis of the target countries of the global activities using the framework models above. The countries under discussion are Finland, Belarus, China, Czech Republic, Hungary, India, Romania, Russia, and Sweden. A detailed report of the comparison is available in (Jaakkola et al, 2011).

The globalization decision in most of the cases was based on the price of work – purely on economic factors. Secondary goals were the availability of special skills or nearness of new clients or new markets. Comparison of the labor price between Finland (100) and some selected countries is introduced in Table 1.

Table 1. Price of work relative to Finland (100)

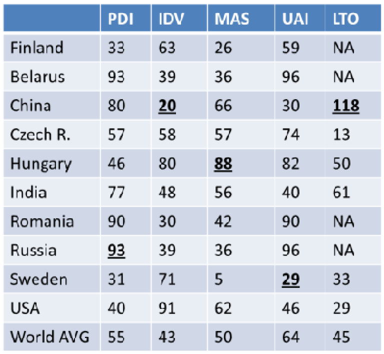
Belarus	China	Czech Republic	Hungary	India	Romania	Russia	Sweden	USA
30	40-50	50-60*	50-80*	30-35	30	30-40	120*	280**

The prices in the table are based on the data given by the case organization of our study when available. The numbers marked with * is estimated by the author on public statistics. The USA value (marked with **) represents the value in an expert organization on the top of organization pyramid. To get full economic benefit on a global site the indirect costs of the cheaper workforce should not exceed the cost of local (Finnish) work. In this context we do not go into a deeper analysis of the economic aspects of globalization.

According to the Hofstede model, the countries concerned get the index score as listed in Table 2. As seen in the table, the differences between countries are remarkable. The *vicinity (nearness) of indices* indicates a good fit from the collaboration point of view and a big difference indicates the opposite situation. The simplified conclusion is based on the expectation that people who understand the culture of others are also more suitable for distributed collaboration. A detailed comparison can be made by analyzing Table 1. The nearness of the scores compared to Finland can be evaluated by counting how many (of the nine) countries are on the same side of the world average as Finland. This comparison gives the following results: PDI 3/9, IDV 4/9, MAS 4/9, UAI 4/9. LTO does not have a full coverage of scores. It is also notable that the distance between minimum and maximum values is reasonably big. The interpretation above is heavily simplified: it is important to notice that the scores are relative and not directly comparable based on their values. The *cultural distance* of two cultures would be defined as the difference of selected cultural properties (in this case index values). The biggest (cultural) difference to Finland is bolded and underlined in the table.

Table 2. Scores of Hofstede indices in selected countries

(Belarus is derived from the profile of Russia; WVS scores are not available)



	PDI	IDV	MAS	UAI	LTO
Finland	33	63	26	59	NA
Belarus	93	39	36	96	NA
China	80	**<u>20</u>**	66	30	**<u>118</u>**
Czech R.	57	58	57	74	13
Hungary	46	80	**<u>88</u>**	82	50
India	77	48	56	40	61
Romania	90	30	42	90	NA
Russia	**<u>93</u>**	39	36	96	NA
Sweden	31	71	5	**<u>29</u>**	33
USA	40	91	62	46	29
World AVG	55	43	50	64	45

Power distance (PDI) differences in the table indicate differences in organizational structure: In India, two managers are needed to manage one expert, in Russia hierarchy beats democracy, etc. Differences in IDV indicate differences in people's basic behavior. Individualistic cultures are able to separate work and leisure time, and in less individualistic cultures, the work society also follows into the leisure time. Feministic cultures (low MAS value) prefer family and leisure time to work. This should be taken into account when motivation factors related to work are discussed. Uncertainty avoidance (UAI) is typical for cultures with a fear of losing face – typical in Eastern cultures; giving feedback and leading an organization is different in high and low UAI cultures. High PDI cultures tend to work according to pre-defined processes supported by complicated documentation (Jaakkola et al., 2011).

Fig. 4 shows the positions of selected countries according to the Lewis model. As seen in the figure, only Sweden is near Finland in the same continuum of the triangle. Cheap labor countries, popular targets in globalization (India and Russia), follow a cultural pattern that is fully based on different basic factors than the Nordic and North European one of Finnish. This creates challenges in organizing the management of the organization and in embedding good leadership practices among the personnel.

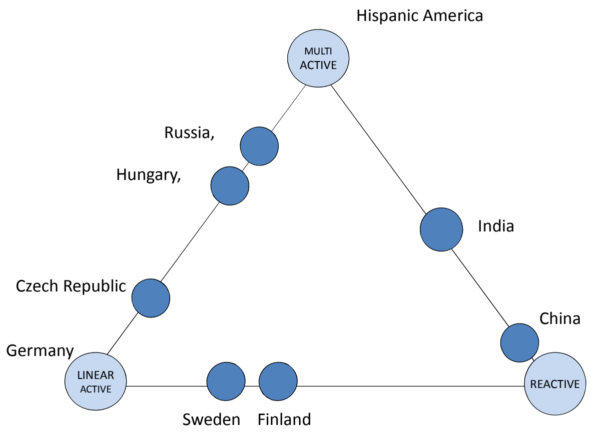


Fig. 4. Selected countries classified according to the Lewis model (Lewis, 2011; modified)

Project managers and team members in linear-active cultures (like Finland) generally demonstrate task orientation. They look for technical competence, place facts before sentiment, logic before emotion; they are deal-orientated, focusing their own attention and that of their staff/team/individuals on immediate achievements and results. They are orderly, stick to agendas and inspire staff with their careful planning. Multi-active project managers and team members are much more extrovert, rely on their eloquence and ability to persuade and use human force as an inspirational factor. They often complete human transactions emotionally, investing their time in developing the contact to the limit. Such project managers and team members are great networkers, working according to people-time rather than clock-time. Project managers and team members in reactive cultures are equally people-oriented but dominate with knowledge, patience, and quiet control. They display modesty and courtesy, despite their accepted seniority. They create a harmonious atmosphere for teamwork. Subtle body language replaces excessive words. They know their companies well (having spent years working in various departments), giving them balance and the ability to react to a network of pressures. They are also paternalistic (Jaakkola and Heimbürger, 2010).

The discussion above would be shortly concluded by introducing the concept cultural distance of countries. In using Hofstede's model it means the difference in score values between the countries and in Lewis' model the distance in the triangle; the Lewis' model also indicates the importance to understand the driving forces in behavioral pattern in the terms related to the corners of the triangle; the distance is interpreted to bigger, if the countries are located in different sides (behind the corner") of the triangle.

4 Culture Sensitivity in Software Engineering

4.1 Related Studies

There is a lack of objective studies of the success factors of globalization in the SE industry. In connection with our research project, the following are recommended for further reading. The article of *Walsham* (2002) provides a theoretical framework for cross-cultural software production and use. The paper reports two case studies, one from *Jamaica* and one from *India*. The framework is based on a structurational analysis method, which is compared to Hofstede's model and the findings of two offshoring cases.

In their article, *Siakas* et al. (1999) deal with attitudes to software quality and Total Quality Management (TQM) in the organization. The result is a framework that can be applied for further analysis. It shows the differences in attitude to quality issues in software engineering. The framework applied is Hofstede's five-dimension model.

The same framework is also applied by *Abraham* (2009) in his conference paper analyzing cultural differences as a part of software life cycle management. Different practices and attitudes in software life cycle management and content of the life cycles are reported. The main findings focus on contracting, organizing the life cycle phases, attitude to working time, meeting practices, teamwork, feedback practices, expectations in communication, and risk management.

The conference paper of *Hawthorne and Perry* (2005) opens up the discussion on SE education to meet the needs of distributed and multicultural organizations. Requirements for SE education are discussed; the role of the architecture and an appropriate modular structure for software are highlighted to resolve the problems encountered.

One of the reports providing very concrete results of SE outsourcing has been published by Krishna and Walsham (2004). They report experiences in outsourced software development in *India*, *Japan* and in some *European countries* by analyzing real outsourcing cases. Differences in agreement culture, level of expected documentation and in the mental mode of the cultures (attitude to bureaucracy, authorities, the role of values and norms etc.) are recognized as sources of problems. The problems arising inside cross-cultural teams are different from those arising inside teams representing the same nationalities / cultures / language groups. The beneficial use of "bridging teams" was also seen as important in unifying the organizational culture in the long term.

The conference paper by *Borchers* (2003) analyzes software development in three cultures, *Japanese*, *Indian*, and *American*. The analysis applies Hofstede's model, focusing on PDI, UAI, and IDV. The differences recognized are in attitudes to work, the role of software architecture, division of work and product management.

A paper by *Simcock* (1998) reports a case of the cultural mix in SE design teams as a part of a project included in a university-level IT curriculum. The teams represented the cultural mix in the role of SE design teams. The members of the teams were undergraduates studying IT. The main finding is that in teams, the cultural strengths of some members support the weaknesses of others.

The articles briefly reviewed above do not answer the question of how to take cultural issues into account in software life cycle processes. However, they point out factors that would be significant for detailed study in the analysis of cultural sensitivity.

4.2 Steps in Multi-Cultural Software Business Globalization

In the introduction of this paper we referred to the STEP project (Steps in Multi-Cultural Software Business Globalization; 2009-2011) and some of the results produced were also listed. In addition to the theses (Karttunen, 2010; Wesslin, 2010; Statkaityte, 2011), several conference and journal papers have been published. A short synthesis of these is worth giving.

One paper (Jaakkola, 2009) introduced a *five-factor model* specifying five viewpoints to globalization in the software industry: organizational issues (**O**), frameworks of culture (**C**), direction of globalization (**D**), ownership (proprietor) of the global organization (**P**), and the artifact produced (**A**). In addition, a literature review – i.e. the results of available knowledge mining – is included in the paper.

The paper by Jaakkola and Heimbürger (2009) includes a deeper analysis of the factors introduced in (Jaakkola, 2009); the approach selected is based on the characteristics of the work. The concept of DCCI work (Distributed, Cross-cultural, Collaborative, Intellectual) is introduced. The D, C, and P factors of the five-factor model are discussed in detail.

Jaakkola et al.(2009) deals with the learning perspective of globalization. Globalization is seen as a context and the global organization is analyzed as a learning organization according to Nonaka's SECI model. The results from the university education point of view are deduced by finding answers to the question "How should globalization be taken into account in adaptive university curriculum content?" The term *Context Aware Software Engineering (CASE)* is used to cover the key results and specify the approach in development work.

Another paper (Jaakkola, Heimbürger et al., 2010) continues the discussion on organizational level adaptation in a multicultural context. At the beginning, the essential terminology related to culture and globalization is discussed. A three-layer model – Process (P), Knowledge context (K), Multicultural Context (M) – is used as a framework to structure the complexity. The organization is seen as a learning organization that must be able to adapt in different situations at all three levels (PKM). The knowledge creation model is applied in selected case situations and cultures. In addition, the different levels and forms of problems related to global organization are discussed.

The theme "learning organization" is also discussed by Jaakkola et al. (2011) in their journal paper. The globalization GRID – different organizational forms of globalization - is revisited (see Jaakkola, 2009) as well as different dimensions of culture. The main contribution is included in the analysis of six company cases representing different forms of globalization. The main findings are categorized and the globalization path is captured in the form of a *Globalization Maturity Model (GMM).* The model introduces the maturity of the organization from the globalization point of view taking various aspects into account.

The productivity perspective in global software engineering (GSE) is analyzed by Aramo-Immonen, Jaakkola et al. (2011). The economic consequences of globalization are not well managed by companies; hidden costs and lower productivity are often forgotten in calculations when the globalization decision is made. The paper analyzes multicultural ICT companies from their productivity perspective through the lens of cultural differences. The paper reports findings based on general cultural studies and reported experiences (the same cases as in Jaakkola et al, 2011) that seem to affect productivity in the software industry.

One of the most difficult problems in global organizations is *trust creation*. This view of global software engineering is taken by Aramo-Immonen, Jaakkola et al. (2011a). Trust building is also seen as one of the key problems in globally operated organizations by Paasivaara et al. (2010). Their paper explores the trust-building processes in global SE from a cultural perspective. Staff from five large multinational SE companies were interviewed (the same as in the other papers introduced). In the conceptual part of the article, firstly, the domain of SE is introduced, secondly, there is a discussion on the concept of trust and trust-building processes, and thirdly, cultural dimensions affecting trust-building processes are examined. The findings from the case companies are discussed at the end of the paper.

A summary, mainly derived from (Jaakkola 2009; Jaakkola and Heimbürger 2009; Jaakkola et al. 2011) the following aspects are worth of noticing:

- Salary benefits in some evaluations are expected to be a temporary phenomenon. There is an expectation that salaries in lower-cost countries will rise faster than those in high-cost countries; filling the gap will not take more than 10 years. The experts' (interviewees of this study) opinion is different – they expect it to take more than one generation.
- The term "*emerging culture*" was used to indicate a country or culture in which the gap between an attainable level and the current level of welfare is larger than that in "*mature cultures*" and countries. The motivating factors in these country categories are different.
- The trend in Finnish-owned overseas sites is moving from a *Resource pool* towards a *Competence Center*, from *fragmented* responsibilities towards *comprehensive* responsibilities, from *separate responsibilities* towards a *role in the global value chain*. The current situation seems to support the hypothesis that work division is based on *front- and back-end processes*, i.e. the client-oriented part of the work is still being done in Finland, and the technical work according to detailed specifications is being carried out at overseas sites. The trends described are used as a basis for a *globalization maturity model* (GMM), which indicates the state of globalization and activities expected to reach the next level in maturity (analogous to the CMMI of software processes).
- Religion has an important role in the behavior of societies – religion is also one of the factors indicating ranks in Hofstede's indices.
- Decision-making in different cultures differs a lot. A high PDI indicates hierarchical decision-making. In business activities this means that agreements cannot be finalized in business meetings. There are also differences in the role of the individual person and organization in business transactions – sometimes

agreements are made between people, sometimes between organizations, and sometimes as a combination.

- Joint features in the cases introduced are the role of universities, and the importance of local ”consultation” to provide contacts to and knowledge of the local administration. In some cases “strategic alliances” between the foreign company and local organizations support success in globalization. Local HR practices and recruiting channels are needed.
- Language problems seem to be the main source of misunderstanding. Special attention to language problems should be given in countries with a strong cultural identity, like Germany and France. Use of common tools and semi-formal specifications support language-free mutual understanding.
- The role of *bridgehead teams* (teams piloting multicultural practices) is important in cultural adaptation. Formal and informal forums for experience exchange are also positive indicators of success.

These results are generalized from the interviews at the six case companies. The companies represent different categories in our globalization GRID (Jaakkola, 2009) and the results are expected to cover a reasonably wide variety of globalization situations. Adding new companies does not seem to bring any significant new aspects to the discussion.

4.3 Culture Sensitivity of Software Life Cycle Processes – Standardization View

As discussed earlier (Section 2) in this paper, the purpose of process models adopted to guide the organization’s software development is to unify the flow and content of the activities implemented. Based on our findings in the multicultural and global software engineering context, unification does not fit the needs of reality. There are several culture-sensitive processes that must be adaptable and “fine-tuned” in the context of usage. Typical processes of this kind include communication and people-oriented activities. Critical points are also transmission points where control of an asset is transferred from one part of the organization to another. Applications of process models in organizations are based on standards – such as ISO/IEC 15504 (2005) and SEI (2011).

One of the first published proposals for adding culture sensitivity to software engineering related process models was made by Biro et al. (2002). They propose a third, cultural, dimension of CMMI/SPICE architecture, in addition to the process and capability dimensions of the existing models (Fig. 5). The authors considered the fact that, despite globalization, there is a considerable difference in local outlook and expectations among global team members. Biro et al. explored the differences in cultural value systems from the software process perspective. In order to characterize different value systems, Hofstede’s work was used as a basis. The authors extended the CMMI model with a third cultural dimension and argued that the extension is valid because national culture influences the effectiveness of various practices.

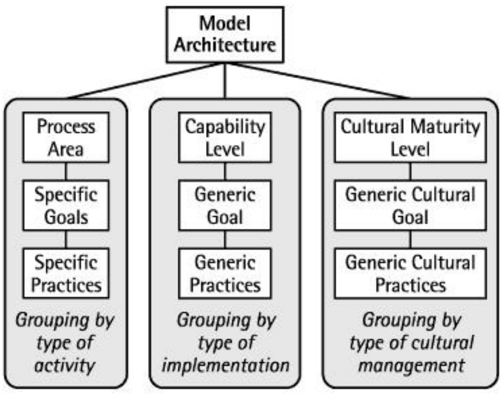


Fig. 5. CMMI extension (Biro et al. 2002)

The maturity levels proposed include:

- *level 0 (closed):* no differences in cultural value systems are allowed;
- *level 1 (open):* open enough to allow differences in cultural value systems;
- *level 2 (model based):* consideration of cultural differences is based on a scientifically established model;
- *level 3 (comprehensive):* the cultural model is comprehensively applied to all specific and generic practices;
- *level 4 (tailored):* applying the cultural model is based on quantitatively managed business needs;
- *level 5 (competency driven):* the cultural model is refined, extended, or fully changed on the basis of competency acquired through quantitatively managed business needs.

As a reference, the same structure was also implemented by Jaakkola et al. (2011) in a wider interpretation (different scopes).

Richardson et al. (2010) observed problems with CMMI and ISO/IEC 15504 in the global environment, even though these process improvement models work well in local environments. The authors identified twenty-five factors to be taken into account when setting up virtual global teams. Furthermore, they developed a software process area, called Global Teaming (GT), similar to the structure of CMMI, with specific goals and sub-practices. These explicitly defined processes can be used as a support mechanism for GSE implementation (Statkaityte, 2011).

Cultural factors are particularly important in software development because they affect global virtual teams (Huang and Trauth, 2007). Processes in software development are complexly interdependent and iterative. Software products are less tangible, and knowledge involved in software development is very tacit and fast-changing in nature. The uncertainty in the software development process requires a lot of communication, both formal and informal (ad hoc). Many activities addressed in software engineering processes are culturally sensitive, i.e. they are carried out by individuals that come from different cultural backgrounds. Example activities that can be found in ISO/IEC 15504 (2005), such as joint activities between customer and supplier, change requests, problem resolution, reporting errors, testing, code reviews, audits, configuration management, team management, risk identification, giving feedback, identifying responsibilities etc. can be carried out in different ways depending on the cultural context. Software development processes involve not only technical skills, but also soft skills through the human resources that perform those processes. Denise (2011) proposes that the complex human interaction in software engineering projects be expressed using the Triple C Model: Communication (information exchange), Cooperation (inside the group), and Coordination (efficiency). Statkaityte (2011) has integrated the model in a modified ISO/IEC 12207 (2008) process model (Fig. 6) to introduce the culture-sensitive features in software engineering related processes.

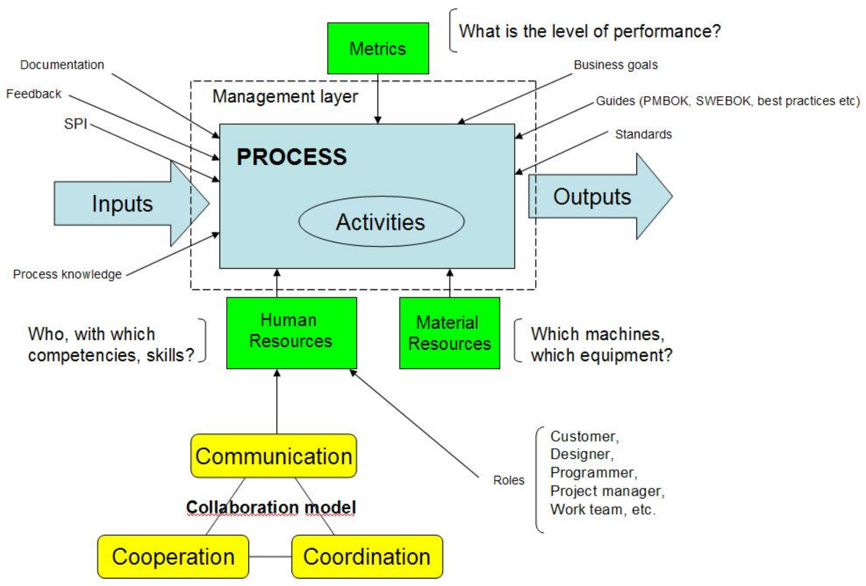


Fig. 6. CCC model integrated in ISO/IEC 12207(2008) process model (Statkaityte, 2011)

Software development involves a lot of collaborating activities, which rely on the effectiveness and quality of communication channels, because miscommunication between client and vendor might cause the failure of the development initiative. Additionally, social and inter-personal aspects among software team members play a major role in software development. In other words communication in software projects is about exchanging messages and negotiating; coordination is about managing people, their activities and resources; cooperation is about working together in a shared workspace. These activities are also the main source of problems in global organizations.

A process is a set of activities that transform inputs into outputs. Cultural sensitivity in SE processes is understanding the ways in which process outcomes or internal process functions change when an SE process is executed by people coming from different cultural backgrounds. Software engineering standards define *What* needs to be done or achieved, the interpretation of *How* it could be achieved depends on the management layer and the people that execute the processes. This is where they uniquely shape the outcomes of the processes due to their different ways of thinking and doing things (Statkaityte 2011).

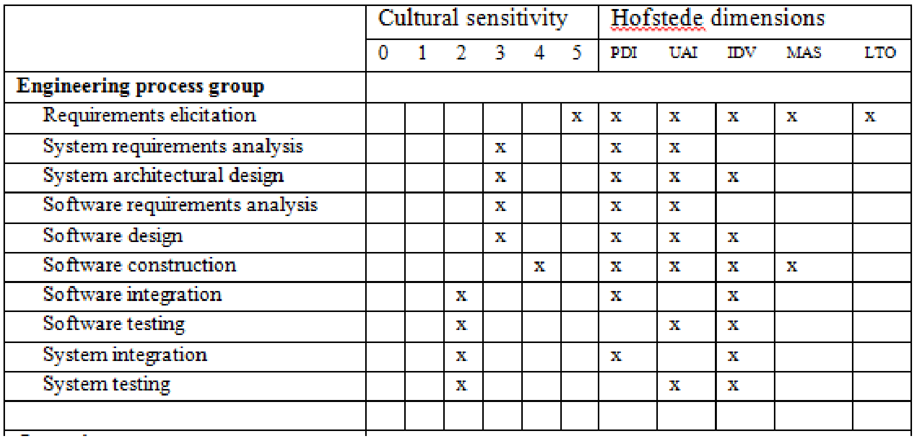
5 The CSAM Analysis Model

As a conclusion and practical result of our work a Cultural Sensitivity Assessment Model (CSAM) is introduced. The ideas introduced in earlier sections of this paper are built into the model. It can be used

- as a tool to improve the processes of an organization to take into account the cultural aspects in the processes of the organization;
- as a framework to understand the importance of the cultural aspects related to the processes (results of the analysis).

The method itself is simple: it integrates the process map of a software organization with Hofstede's cultural dimensions. Each process is assessed to take into account the cultural aspects (see sub-section 4.3). The theoretical framework of the tool has been explained in earlier sections of this paper.

Table 3. Results of culture sensitivity analysis of selected processes using the CSAM method



	Cultural sensitivity						Hofstede dimensions				
	0	1	2	3	4	5	PDI	UAI	IDV	MAS	LTO
Engineering process group											
Requirements elicitation						x	x	x	x	x	x
System requirements analysis				x			x	x			
System architectural design				x			x	x	x		
Software requirements analysis				x			x	x			
Software design				x			x	x	x		
Software construction					x		x	x	x	x	
Software integration			x				x		x		
Software testing			x					x	x		
System integration			x				x		x		
System testing			x					x	x		

Table 3 illustrates the results of the sensitivity assessment made by Statkaityte (2011). The analysis (in this example) is focused on the ENG process group of the process map used by ISO/IEC 15504 (2005) (adapted from ISO/IEC 12207, 2004).

The analysis method includes a questionnaire that introduces the concepts related to the process structure of an organization, the capability levels of processes, and the cultural dimensions measured. The interviewees are asked to give an expert opinion on the culture sensitivity (rating 1-5) of the processes and the elements of culture sensitivity using Hofstede's dimensions. The results are interpreted by the reviewer and illustrated in table format (as in Table 3). For example, in this evaluation Requirements Elicitation is recognized as very culture-sensitive and the sensitivity is related to all Hofstede's dimensions. The outcomes of this process determine the success or failure of the software project. The elicitation process requires a great deal of communication between vendor and customer. This is also the process where knowing the cultural differences between vendor and customer gives a competitive advantage. The significance of Hofstede's indices related to the process assessed is marked by (x). The results would be even more informative if the *strength (importance)* of the index could be categorized. An illustrative solution would be the use of colors: red=very important, yellow=worth noting, green=some importance, white=not worth taking into account.

A high Power Distance Index (PDI) contributes towards hierarchical and bureaucratic forms of decision-making and communication processes. For instance, in Thailand decision-making takes longer, as every stage has to be reported to management for final decisions. The person's position in the hierarchy might appear to be more important than technical issues.

UAI deals with tolerance for ambiguity and uncertainty. Indians (low UAI) work with the understanding that there will be frequent requirement changes. They also accept ambiguous statements and implicit understanding of concepts. Japanese (high UAI), on the other hand, being very structured and organized teams, finalize the requirements only after a longer and thorough requirements analysis phase.

Individualism vs. collectivism (IDV) could be illustrated in customer-vendor relationships. People from collectivistic cultures (India, China) want to build strong social relationships. People that come from individualistic cultures (USA) might see it as too much time spent on building unnecessary relationships, and collectivistic people might think that individualistic people do not wish to settle with the group.

The Long-Term Orientation versus Short-Term Orientation (LTO) dimension is about investing far into the future as opposed to the goal of achieving quick results. One theoretical conflict that could arise is if the client is from a short-term oriented culture and the vendor comes from a long-term oriented culture. Whereas the client would like to see some results very fast, the vendor might feel that the deadlines are too tight, because the vendor would like to take more time for understanding, agreeing and prioritizing user requirements rather than starting the coding right away.

The results in Table 3 are based on the interviews in our case companies and validated with them. The thesis of Statkaityte (2011) cover all process groups of the process model framework. On a general level, the results are suggestive and based on the heuristic / subjective opinions of the interviewer (when the results are interpreted) and the interviewee (when interviewed). These could also be extended with the elements from the related studies and other company cases. It is important that the basic concepts are clear, well specified and understood. It is possible to include the analysis in a traditional process assessment, which is also based on assessment questionnaires applied at different capability levels.

6 Conclusions and Further Work

6.1 Summary of the Results

The purpose of the paper is to introduce a means for a better understanding of the role of cultural differences in multicultural software development work. A typical approach in software companies is to follow carefully the processes defined by their quality management system. The paradox is that the higher the maturity of the organization, the more unified its processes throughout the organization. This approach is effective in the case of homogenous organizations. In the case of heterogeneous organizations problems may occur; one of the sources of the problems may be the cultural differences between the employees.

Communication-related issues are seen as one of the most difficult to address in an organization. In distributed organizations, face-to-face communication is difficult. The level of difficulty increases as the geographical distance grows; quite often additional elements are the different time zones and differences in cultural background. A lot of problems may be avoided by using well selected and widely adopted tools; there is no way to dissipate the cultural background, not even when the person is moved from one global unit of the organization to the same location as other team members. Culture is partially built in the genes, partially learned and partially adopted and adapted by the individual (see the writings of Hofstede and Lewis referred to in this paper in several contexts). In the case where the organization wants to be effective and (1) to avoid the problems caused by cultural collisions and (2) turn the cultural differences into a benefit, it must recognize the source of the problems.

Three sections after the introduction of this paper provide the cornerstones to the last section: an introduction to the software life cycle process structure and to frameworks of cultural analysis. Section 4 introduced the related work available in the literature and a synthesis of the research work made by the author and his research group on this topic. These results have been further "cultivated" to find an analysis tool that is useful in analyzing the culture sensitivity of software processes in Section 5. The work is still ongoing and this paper is based on a conceptual introduction rather than a presentation of crystallized and final results. Separate papers have approached the same problem from different viewpoints; synthesis of these is a part of our future work.

6.2 Future Work – An Alternative Approach

In our studies we have approached the problems related to the multicultural organizations from case and organizational point of view. On messages built-in into this paper is to notice the importance of communication between the members of the teams. The source of problems is mainly in communication; in homogenous teams the probability of misunderstanding is lower than in heterogeneous ones. Communication is based on concepts that are communicated by speaking, using written text, using semiformal specifications, etc.

One interesting and easy approach to implementing a simple sensitivity analysis related to software engineering processes is to use a lexical analysis of the process descriptions. This would be used either as a rough analysis tool (easy to automate) or to complete the interview-based results either as a pre-study or as a complementary study. ISO/IEC 15504 (2005) specifies the process elements as explained in Section 2 of this paper: Purpose, Outcomes, Base Practices, Work Products. The purpose of Requirements Elicitation process is defined as follows:

The **Purpose** of Requirements Elicitation is to gather, process, and track evolving customer needs and requirements throughout the life of the product and/or service so as to establish a *requirements baseline* that serves as the basis for defining the required work products. Requirements Elicitation may be performed by the acquirer or the developer of the system.

The **Outcomes** are defined: As a result of successful implementation of Requirements Elicitation:

1) continuing communication with the customer is established;
2) agreed customer requirements are defined and *baselined*;
3) a change mechanism is established to evaluate and incorporate changes to customer requirements in the baselined requirements based on changing customer needs;
4) a mechanism is established for the continuous monitoring of customer needs;
5) a mechanism is established to ensure that the customers can easily determine the status and disposition of their requests; and
6) enhancements arising from changing technology and customer needs are identified and their impact managed.

Base practices that advance satisfaction of the outcomes are:

BP1- Obtain customer requirements and requests. Obtain and define customer requirements and requests through direct and continuous solicitation of customer and user input.

BP2- Understand customer expectations. Ensure that both supplier and customer understand each requirement in the same way. Review with customers their requirements and requests to better understand their needs and expectations and to check the feasibility and appropriateness of their requirements.

BP3- Agree on requirements. Obtain agreement across teams on the customer requirements, obtaining the appropriate sign-offs by representatives of all teams and other parties contractually bound to work to these requirements.

BP4- Establish customer requirements baseline. *Formalize* the customer requirements and establish as a baseline for project use and monitoring against customer needs.

BP5- *Manage* customer requirements changes. *Manage all changes* made to the customer requirements against the customer requirements baseline to ensure enhancements resulting from changing technology and customer needs are identified and that those who are affected by the changes are able to assess the impact and risks and initiate appropriate change control and risk mitigation actions.

BP6- Establish a customer query mechanism. Provide a means by which the customer can be aware of the status and disposition of their requirements changes.

The text above includes the definition of the Requirements Elicitation process. The underlined words refer to communication and team-oriented activities, in which the role of human-to-human interaction is essential and also a typical source of problems. The words in *italics* refer to the activities that are managed by predefined practices and / or tools. The latter are not culture-sensitive components in the process.

This "method" is not systematically tested and the results are not. Like the results of CSAM analysis, even these results may be interpreted to include a lot of subjective elements. Its objectivity may be improved by including more context related "intelligence" in the analysis – this would be also a task for further investigation.

References

1. Abraham, L.R.: Cultural Differences in Software Engineering. In: Proceedings of the Second Annual Conference on India Software Engineering, Pune, India, pp. 95–100 (2009)
2. Aramo-Immonen, H., Jaakkola, H., Keto, H.: Multicultural software Development: The Productivity Perspective. International Journal of Information Technology Project Management (IJITPM) 2(1), 19–36 (2011)
3. Aramo-Immonen, H., Jaakkola, H., Linna, P.: Trust Creation in Multi-cultural organisations. Journal of Global Information Technology Management (2011)
4. Biro, M., Messnarz, R., Davison, A.G.: The Impact of National Cultural Factors on the Effectiveness of Process Improvement methods: The Third Dimension. ASQ 4(4) (2002), `http://asq.org/pub/sqp/past/vol4_issue4/biro.html`
5. Borcheres, G.: The Software Engineering Impacts of Cultural Factors on Multi-cultural Software Development Teams. In: Proceedings of the 25th International Conference on Software Engineering (ICSE 2003), pp. 540–545. IEEE (2003)
6. Crosby, P.B.: Quality is Free. McGraw Hill, New York (1979)
7. Denise, L.: Collaboration vs. C-Three (Cooperation, Coordination, and Communication). The Rensselaerville Institute 7,3 (2011), `http://www.ride.ri.gov/adulteducation/Documents/Tri%20part%201/Collaboration%20vs.%20the%203c%27s.pdf`
8. Hawthorne, M.J., Perry, D.E.: Software Engineering Education in the Era of Outsourcing, Distributed Development, and Open Source Software: Challenges and Opportunities. In: Inverardi, P., Jazayeri, M. (eds.) ICSE 2005. LNCS, vol. 4309, pp. 166–185. Springer, Heidelberg (2006)
9. Hofstede, G., Hofstede, G.J.: Cultures and Organizations - Software of the Mind: Intercultural Cooperation and Its Importance for Survival, 1st edn. McGraw-Hill, New York (2005)
10. Hofstede, G., Hofstede, G.J., Minkov, M.: Cultures and Organizations- Software of the Mind: Intercultural Cooperation and Its Importance for Survival, 3rd edn. McGraw-Hill, New York (2010)
11. Hofstede, G.: Cultural Dimensions. Geert Hofstede'sresource pages of cultures (2011), `http://www.geert-hofstede.com`
12. Huang, H., Trauth, E.: Cultural Influences and Globally Distributed Information Systems Development: Experiences from Chinese IT Professionals. ACM portal (2007), `http://portal.acm.org/citation.cfm?id=1235008`
13. Humphrey, W.: Managing the Software Process. Addison Wesley (1989)

14. ISO/IEC 12207/Amd2:2004: Systems and Software Engineering - Software life cycle processes - Amendment 2 (2004)
15. ISO/IEC 12207: Systems and software engineering - Software life cycle processes. ISO (2008)
16. ISO/IEC 15504-5: Information technology – Process Assessment – Part 5: An exemplar Process Assessment Model. ISO (2005)
17. itSMFInternational:The IT Service Management Forum (2011), `http://www.itsmfi.org/`
18. Jaakkola, H.: Towards a Globalized Software Industry. Acta Polytechnica Hungarica 6(5), 69–84 (2009)
19. Jaakkola, H., Heimbürger, A., Henno, J.: The Roles of Knowledge and Context in Context-Aware Software Engineering - in Terms of Education and Communication. In: Cicin-Sain, M., Prstacic, I.T., Sluganovic, I., Uroda, I. (eds.) MIPRO Conference, pp. 224–230. MIPRO and IEEE, Opatija, Croatia (2009)
20. Jaakkola, H., Heimbürger, A.: Cross-Cultural Software Engineering. Informatologia 42(4), 256–264 (2009)
21. Jaakkola, H., Heimbürger, A., Linna, P.: Knowledge Oriented Software Engineering Process in Multi-Cultural Context. Software Quality Journal 18(2), 299–319 (2010)
22. Jaakkola, H., Henno, J., Linna, P.: From Local to Global - Path towards Multicultural Software Engineering. International Journal of Knowledge and Learning, IJKL (2011)
23. Jaakkola, H., Thalheim, B.: Software Quality and Life Cycles. In: Eder, J., Haav, H.-M., Kalja, A., Penjam, J. (eds.) Advances in Databases and Information Systems, ADBIS 2005, pp. 208–220. Tallinn University of Technology Press, Tallinn (2005)
24. Jaakkola, H., Thalheim, B.: Framework for high-quality software design and development: a systematic approach. IET Software 4(2), 105–118 (2010)
25. Jaakkola, H., Thalheim, B.: Architecture-Driven Modelling Methodologies. In: Heimbürger, A., Kiyoki, Y., Tokuda, T., Yoshida, N. (eds.) Information Modelling and Knowledge Bases XXII, pp. 97–116. IOS Press, Amsterdam (2011)
26. Karttunen, E.: Producing Software Services in a Multi-cultural Environment. MSc Thesis, Information Technology, Pori. Tampere, Tampere University of Technology, Tampere (2010)
27. Krishna, S., Sahay, S., Walsham, G.: Managing Cross-Cultural Issues in Global Software Outsourcing. Communications of the ACM 47(4), 62–66 (2004)
28. Lewis, R.D.: When Cultures Collide. Managing Successfully Across Cultures. Nicholas Brealey Publishing, London (1999)
29. Lewis, R.D.: Cross-Culture – The Lewis Model. Richard Lewis Communications (2000), `http://faculty.fuqua.duke.edu/ciber/ice/Cross%20Culture%20The%20Lewis%20Model.pdf`
30. Lewis, R.D.: Cultureactive – The web based global cultural database. Resource pages of Richard Lewis (2011), `http://www.cultureactive.com`
31. Paasivaara, M., Hiort af Ornäs, N., Hynninen, P., Lassenius, C., Niinimäki, T., Piri, A.: Practical guide to managing distributed software development projects. Aalto University, School of Science and Technology, Espoo (2010)
32. Pfleeger, S.L., Atlee, J.M.: Software Engineering: Theory and Practice, 3rd edn. Pearson Education International (2006)
33. Richardson, I., Casey, V., Burton, J., McCaffery, F.: Global Software Engineering: A Software Process Approach. In: Mistrik, I., Grundy, J., van der Hoek, A., Whitehead, J. (eds.) Collaborative Software Engineering. Springer, Heidelberg (2010), `http://www.springerlink.com/content/k680115651r231w4/`

34. Royce, W.: Managing the Development of Large Software Systems. In: Proceedings of IEEE WESCON, pp. 1–9 (August 1970), http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf
35. SEI - Software Engineering Institute: CMMI Version 1.3 Information Center (2011), http://www.sei.cmu.edu/cmmi/tools/cmmiv1-3/
36. Siakas, K.V., Georgiadou, E., Sadler, C.: Software Quality Management from a Cross-Cultural Viewpoint. Software Quality Journal 8(2), 85–95 (1999)
37. Simcock, A.L.: Does a Multicultural Mix Bring an Extra Dimension to Software Engineering Design Teams? Global Journal of Engineering Education 2(3), 263–270 (1998)
38. Walsham, G.: Cross-Cultural Software Production and Use: A Structurational Analysis. MIS Quarterly 26(4), 359–380 (2002)
39. Statkaityte, R.: Multicultural issues in software engineering processes. MSc Thesis, Information Technology Pori. Tampere University of Technology, Tampere (2011)
40. Wesslin, V.: Globalization practices of software companies from cultural point of view. MSc Thesis, Information Technology, Pori. Tampere University of Technology, Tampere (2010)

Cross-Cultural Multimedia Computing with Impression-Based Semantic Spaces

Yasushi Kiyoki, Shiori Sasaki, Nhung Nguyen Trang, and Nguyen Thi Ngoc Diep

Graduate School of Media and Governance, Keio University, SFC
5322 Endo, Fujisawa, Kanagawa, 252-0816, Japan
kiyoki@sfc.keio.ac.jp
www.mdbl.sfc.keio.ac.jp

Abstract. Over the past decade, the rapid progress of multimedia data management technology has realized the large scale of media data transfer and resource-accumulation in the world. The multimedia computing technology has also been creating new information provision environments in the world-wide scope. Innovative integrations of large scale multimedia data management and computing technology will lead to a new information society.

In the design of multimedia systems, one of the most important issues is how to search and analyze media data (images, music, movies and documents), according to impressions and contexts. We have proposed and introduced a *"Kansei"* and semantic associative search method based on our “Mathematical Model of Meaning (MMM) [11], [13], [14]”. The concept of *"Kansei"* includes several meanings on sensitive recognition, such as "impression", "human senses", "feelings", "sensitivity", "psychological reaction" and "physiological reaction". This model realizes *"Kansei"* processing and semantic associative search for media data, according to user's impressions and contexts. This model is applied to compute semantic correlations between keywords, images, music and documents dynamically in a context-dependent way.

The main feature of this model is to realize semantic associative search in the 2000 dimensional orthogonal semantic space with semantic projection functions. This space is created for dynamically computing semantic equivalence or similarity between keywords and media data.

We have constructed "Cross-Cultural Multimedia Computing Systems” for sharing and analyzing different cultures with semantic associative functions applied to “cultural & multimedia data,” as a new platform of cross-cultural collaborative environments. This environment enables to create a remote, interactive and real-time cultural and academic research exchange among different countries and cultures.

1 Introduction

The field of “*Kansei*” information was originally introduced as the word “aesthetics” by Baumgrarten in 1750. The aesthetics of Baumgrarten had been established and succeeded by Kant with his ideological aesthetics [4]. In the research field of ubiquitous & multimedia systems, it is becoming important to deal with "*Kansei*" information for defining and extracting media data according to impressions and senses of individual users.

In the design of the "*Kansei*" information for media data, the important issues are how to define and represent the metadata of media data and how to search media data dynamically, according to impressions and media data contents. Creation and manipulation methods of metadata for media data have been summarized in [4] and [22].

As a semantic associative search method for multimedia database systems dealing with "Kasei" information, we have proposed the Mathematical Model of Meaning (MMM) [11, 13, 14]. The MMM is a basic model for realizing a semantic associative search method with context recognition mechanisms for computing semantic distances and correlations between different media data, information resources and words. One of the important applications, we have presented a semantic associative search system for images and music [8, 9, 10, 15, 17, 20, 21, 25]. The important feature of this model is that the data objects in databases are mapped into an orthogonal semantic space and extracted by a semantic associative search mechanism. This method realizes the computational machinery for recognizing the meaning of a keyword according to a context (context words) and obtaining the related data objects to the keyword in the given context.

The MMM is applied to a semantic image and music search, as a fundamental framework for representing the metadata and searching images and music. The main feature of this model is that the semantic associative search is performed unambiguously and dynamically in the orthogonal semantic space. This space is created for computing semantic equivalence or similarity between user's impression and image's metadata items which represent the features of image data. We point out that context recognition is essentially needed for multimedia information retrieval. The meaning of information is determined by the relation between contents and the context. The machinery for realizing dynamic context recognition is essentially important for multimedia information acquisition.

The advantages and original points of the MMM are as follows:

(1) The semantic associative media search based on semantic computation for words is realized by a mathematical approach. This media search method surpasses the search methods which use pattern matching for associative search. Users can use their own words for representing impression and data contents for media retrieval, and do not need to know how the metadata of media data of retrieval candidates are characterized in databases.

(2) Dynamic context recognition is realized using a mathematical foundation. The context recognition can be used for obtaining multimedia information by giving the user's impression and the contents of the information as a context. A semantic space is

created as a space for representing various contexts which correspond to its subspaces. A context is recognized by the computation for selecting a subspace. In MMM, the number of phases of contexts is almost infinite (currently 2^{2000} in the general English word space and 2^{180} in the color-image space, approximately). For semantic associative computations of "*Kansei*" information in MMM, we have constructed several actual semantic spaces, such as the general English-word space in 2115 dimensions, the color-image space in 183 dimensions based on [18], and music space in 8 dimensions in the current implementations.

We have applied this method to several multimedia database applications, such as image and music database search by impressionistic classification. We have introduced these research results in [11], [15] and [17]. Through these studies, we have created a new meta-level knowledge base environment by applying those methods to data retrieval, data integration and data mining [12, 16].

In this paper, we present a new application of MMM to "Cross-Cultural Multimedia System" realizing media data selection and exchange among different countries and cultures.

2 An Overview of the Semantic Associative Search Method

In this section, the outline of our semantic associative search method based on the mathematical model of meaning (MMM) is briefly reviewed. This model has been presented in [11], [13] and [14] in detail.

The semantic associative search method in MMM is used to extract information resources corresponding to the words, which represent the user's impression and data contents. Each information resource is mapped in the orthogonal semantic space. This space is referred to as "orthogonal metadata space" or "metadata space." The mathematical model of meaning is used to create the orthogonal metadata space. The mathematical model of meaning gives the machinery for extracting the associated information according to the context.

In extracting appropriate information resources, context words that represent the user's impression and data contents are given as the context. According to these context words, a semantic subspace is selected dynamically. Then, the most related information resource to the context is extracted in the semantic subspace. Metadata are classified into three different types. The first metadata type is used to create an orthogonal metadata space, as a search space for semantic associative search. This type of metadata is referred to as "metadata for space creation."

The second type of metadata is used to express metadata of multimedia information resources (image, music, text and video), which are the candidates for semantic associative search. This type of metadata is referred to as "metadata for information resources."

The third metadata type is used to express a context, which represents user's imagination and impression in semantic associative search. This type of metadata is referred to as "metadata for contexts."

The basic functions and metadata structures in MMM are summarized as follows:

(1) A set of *m* words is given, and each word is characterized by *n* features. That is, an *m* by *n* matrix is given as the data matrix with "metadata for space creation."
(2) The correlation matrix with respect to the *n* features is constructed from the data matrix. Then, the eigenvalue decomposition of the correlation matrix is computed and the eigenvectors are normalized. And, the orthogonal semantic space is created as the span of the eigenvectors which correspond to nonzero eigenvalues.
(3) Images and context words are characterized by using the specific features (words) and representing them as vectors.
(4) The multimedia information resources (image, music, text and video), and context words are mapped into the orthogonal semantic space by computing the Fourier expansion to their corresponding vectors.
(5) A set of all the projections from the orthogonal semantic space to the invariant subspaces (eigen spaces) is defined. Each subspace represents a phase of meaning, and it corresponds to a context or situation.
(6) A subspace of the orthogonal semantic space is selected according to the user's impression, imagination, or the multimedia contents, which are given as a context represented by a sequence of words.
(7) The most correlated multimedia information resource to the given context is extracted in the selected subspace by selecting and applying one of the metrics defined in the semantic space.

3 Cross-Cultural Computing System for Music

This section introduces a cross-cultural computing system for music, which is realized by applying MMM to “cultural-music resources.” We have designed this system to promote cross-cultural understanding and communication by using cultural music. The system consists of music analysis, search and visualization functions, characterized by three main features: (1) a culture-dependent semantic metadata extraction method, which extracts both musical elements (e.g., key, pitch, tempo) and impression metadata (e.g., sad, happy, dreamy) corresponding to properties of each musical-culture, (2) a cross-cultural computing mechanism to represent differences and similarities among various music-cultures, and (3) easy-to-use interfaces designed for helping users to join the music database creation process. The significant feature of our cross-cultural computing system is its multimedia database technology applying of “*Kansei*” impressions, to compute the cultural differences. This system extracts features of music-cultures and expresses cultural-dependent impressions by interpreting cultural-music pieces in the semantic music-space, and makes it possible to compare cultural difference and similarity in terms of impressions among various cultural music resources.

The important objective of this cross-cultural computing system is to evoke impressions and imaginations including the cultural diversity by representing various impression-based responses to music resources from different cultures. There are two main scenarios designed to allow users to attain impressions and imaginations: (1) how a music piece would be interpreted among different music-cultures and (2) how an impression would be composed in different music-cultures. The system realizes metadata extraction, search, visualization, and search functions which have been designed in a culture-oriented way. Two music-domains, impressions (e.g., sad, dreamy and happy) and musical elements (e.g., key, pitch and tempo) are utilized to compare cultural-differences. In this system implementation, it is important how to deal with semantic heterogeneity when impressions are variously expressed among different music-cultures. This system provides a culture-dependent impression metadata extraction method to tackle this challenge with participation of users.

Here, "culture" is defined as the collective knowledge which distinguishes the members of one culture group (human group) from another group's [7]. In addition, we consider "cross-cultural computing" as the similarity and difference computation in each domain belonged to common cultural determinants among several cultures. We also define the term "impression" as the culture-dependent emotion-based response which people of a culture react to any music piece (e.g., sad, dreamy and happy).

In the system implementation, we use traditional music as common cultural determinant. Firstly, music is a powerful medium to express human emotion [2], and some data show that many people are using traditional music as the main means to discover a new culture [19]. In addition, Brown in [1] also asserts that music likely has been a main contributor to reinforcing "groupishness" by offering the opportunities to formalize and maintain group identity. That is, music in a given culture, called music-culture, can become a vital part to formalize the "culture identity".

3.1 System Architecture

The cross-cultural computing system for music consists of eight main functions, as shown in Figure 1, and we have shown the system architecture in detail in [25]. Those functions are grouped into five layers including multimedia databases, metadata generation, search, visualization, and user interface. The overview of each function is described as follows:

(F-1) Musical Element Analyzer extracts music metadata of six basic musical elements (key, tempo, pitch, rhythm, harmony and melody) from music MIDI files.

(F-2) Culture-dependent Impression Metadata Extractor extracts weighted impression words (e.g. sad, happy) of a music piece as metadata from culture-dependent musical elements-impression E-I matrices, created by (F-3) and (F-4), and the elements' values computed in (F-1).

(F-3) Culture-dependent E-I Matrix Generator is a matrix creation function. This function creates a specific matrix representing the relationship between musical elements and impressions of a particular musical-culture by using music samples and filtering functions (Nguyen, N. *et al.*, 2010).

(F-4) Dynamic Culture-dependent E-I Matrix Generator is an extending part of (F-3). This function is to automatically provide a prototype of an E-I matrix from a set of culture-based music files and provide an easy-to-use interface allowing users to amend this matrix.

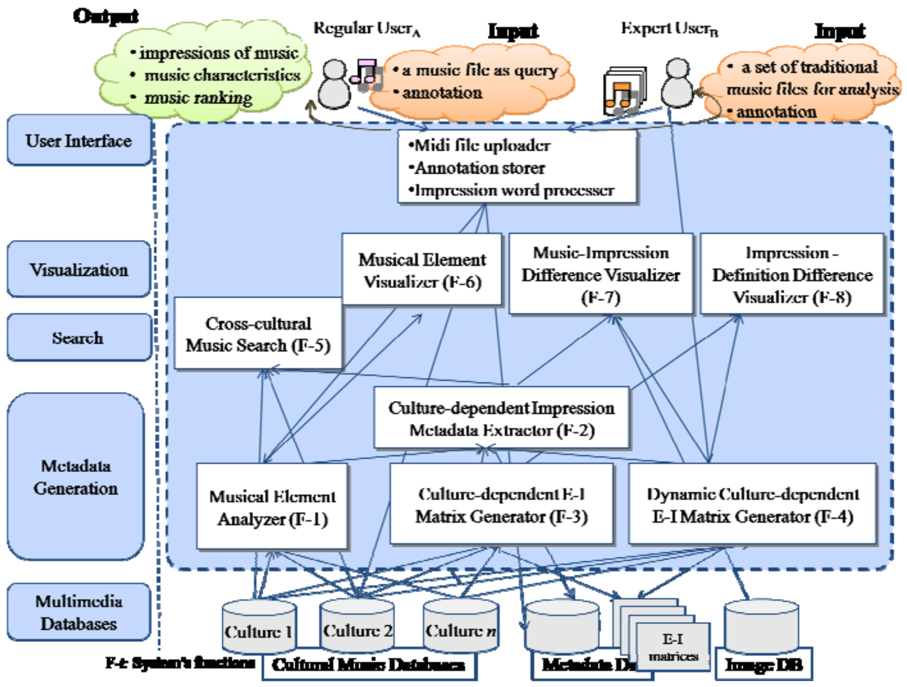

Output
• impressions of music
• music characteristics
• music ranking
Regular User$_A$
Input
• a music file as query
• annotation
Expert User$_B$
Input
• a set of traditional music files for analysis
• annotation
User Interface
•Midi file uploader
•Annotation storer
•Impression word processer
Visualization
Musical Element Visualizer (F-6)
Music-Impression Difference Visualizer (F-7)
Impression - Definition Difference Visualizer (F-8)
Search
Cross-cultural Music Search (F-5)
Culture-dependent Impression Metadata Extractor (F-2)
Metadata Generation
Musical Element Analyzer (F-1)
Culture-dependent E-I Matrix Generator (F-3)
Dynamic Culture-dependent E-I Matrix Generator (F-4)
Multimedia Databases
Culture 1
Culture 2
Culture n
E-I matrices
F-x: System's functions
Cultural Music Databases
Metadata DB
Image DB

Fig. 1. System architecture of our culture-oriented music analysis system

(F-5) Cross-cultural Music Search calculates the correlations between impression, expressed as a context-query given by keywords or music, and impressions of music pieces in cultural music databases by applying MMM as the impression-based semantic associative computing system with a cross-cultural music space. And then, this function provides music ranking as a music-search result, according to correlations between the context query and the impressions of music pieces.

(F-6) Musical Elements Visualizer provides the visualization of musical elements of any music piece which is uploaded by users.

(F-7) Music-impression Difference Visualizer displays various impressions of music pieces; these impressions are analyzed from the viewpoints of different musical-cultures. Image data are also integrated to support users in understanding the diverse impressions.

(F-8) Impression-definition Difference Visualizer shows the diversity of musical properties (musical elements) to express a particular impression (e.g. sad, happy) among different musical-cultures.

3.2 Impression-Based Metadata Extraction for a Cross-Cultural Music Environment

In this subsection, we present a semantic metadata extraction method for a cross-cultural music environment, the key technology in our system (Figure 3). Basic ideas for extracting cultural features in music are as follows: (1) we consider Western classical music as a part of cross-cultural music environment, and (2) we create our culture-dependent impression extraction method for a cross-cultural music environment by extending an automatic music metadata extraction method for Western classical music [10].

We apply the music-analysis method [10] to embark on our method for extracting cultural music features. This method shown in Figure 2 extracts six musical elements of each music piece (key, tempo, pitch, rhythm, harmony, melody) then converts them to impression words (e.g. sad, happy) by using a lexico-media transformation matrix *T* based on the music psychological research of Hevner [5, 6]. Hevner has proposed eight categories of adjectives to represent music impressions (dignified, sad, dreamy, serene, graceful, happy, exciting, and vigorous) and she also devised a correlation table between these adjective groups and six musical elements. However, Hevner's table is suitable to only Western classical music. For creating a cross-cultural environment, as shown in Figure 3, we have designed a culture-dependent metadata extraction as follows: (1) we adopt the schema of Hevner's table and set it as a unified schema to create a table expressing musical elements-impressions relationship in database and (2) we create our process to express specific musical elements–impressions in the E-I transformation matrix *T* for each musical-culture by using music samples and filtering functions [25]. Hereinafter, we call this matrix *T* as E-I matrix *T*.

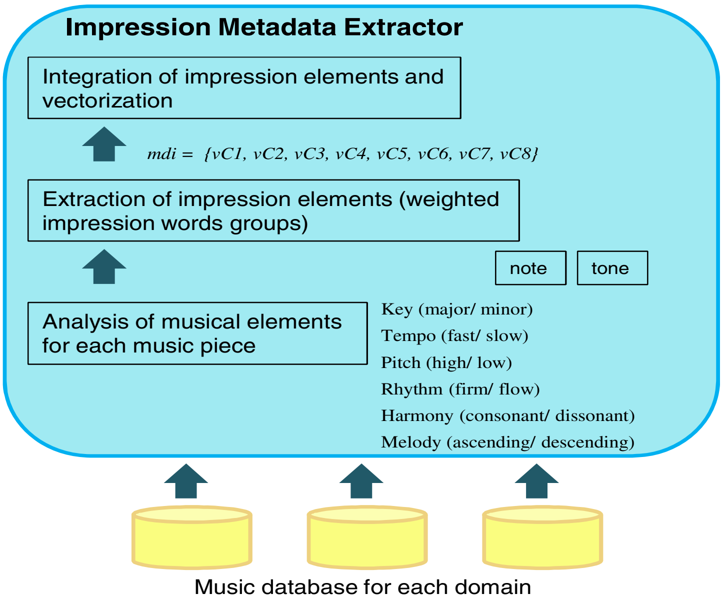

Impression Metadata Extractor
Integration of impression elements and vectorization
mdi = {vC1, vC2, vC3, vC4, vC5, vC6, vC7, vC8}
Extraction of impression elements (weighted impression words groups)
note
tone
Analysis of musical elements for each music piece
Key (major/ minor)
Tempo (fast/ slow)
Pitch (high/ low)
Rhythm (firm/ flow)
Harmony (consonant/ dissonant)
Melody (ascending/ descending)
Music database for each domain

Fig. 2. The impression metadata extraction method in music

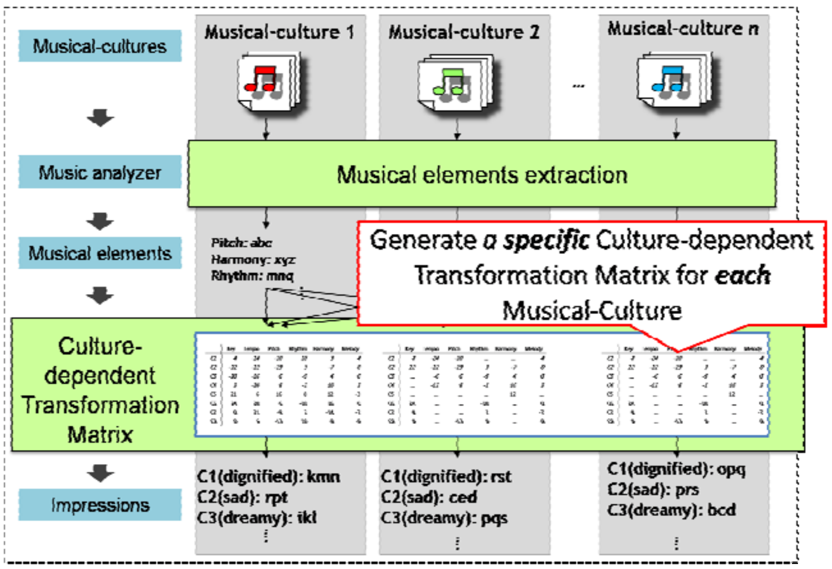


Fig. 3. The cross-cultural music environment for extracting cultural-dependent impressions from cultural-music resources

Our cross-cultural computing system for music is a novel platform to evoke users' imaginations and impressions for the cultural diversity by presenting the variety of emotional responses to music from different cultures. This system makes it possible to realize universal impression analysis to various music-cultures sharing various music-properties.

4 The 5D World Map System with Semantic and Spatiotemporal Analyzers Applied to Cross-Cultural Multimedia Computing

In this section, we introduce the architecture of a multi-visualized and dynamic knowledge representation system "5D World Map System [20,21]," applied to cross-cultural multimedia computing. The basic space of this system consists of a temporal (1st dimension), spatial (2nd, 3rd and 4th dimensions) and semantic dimensions (5th dimension, representing a large-scale and multiple-dimensional semantic space that is based on the Mathematical Model of Meaning (MMM)). This space memorizes and recalls various multimedia information resources with temporal, spatial and semantic correlation computing functions, and realizes a 5D World Map for dynamically creating temporal-spatial and semantic multiple views applied for various "cross-cultural multimedia information resources."

We apply the dynamic evaluation and mapping functions of multiple views of temporal-spatial metrics, and integrate the results of semantic evaluation to analyze cross-cultural multimedia information resources. MMM is applied as a semantic associative search method [11,13] for realizing the concept that "semantics" and "impressions" of cultural multimedia information resources, according to the "context". The main feature of this system is to create world-wide and global maps and views of cultural features expressed in cultural multimedia information resources (image, music, text and video) dynamically, according to user's viewpoints. Spatially, temporally, semantically and impressionably evaluated and analyzed cultural multimedia information resources are mapped onto a 5D time-series multi-geographical space. The basic concept of the 5D World Map System is shown in Figure 4.

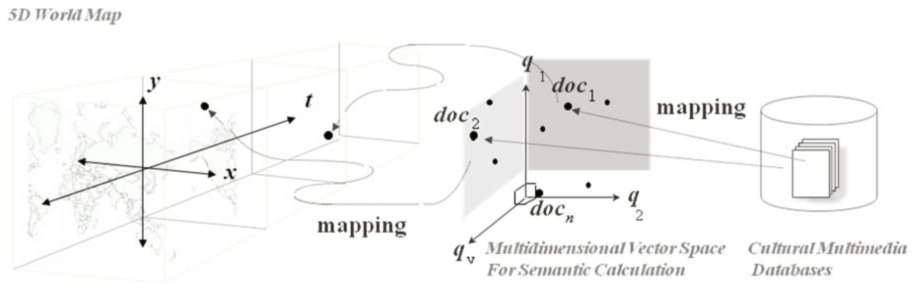
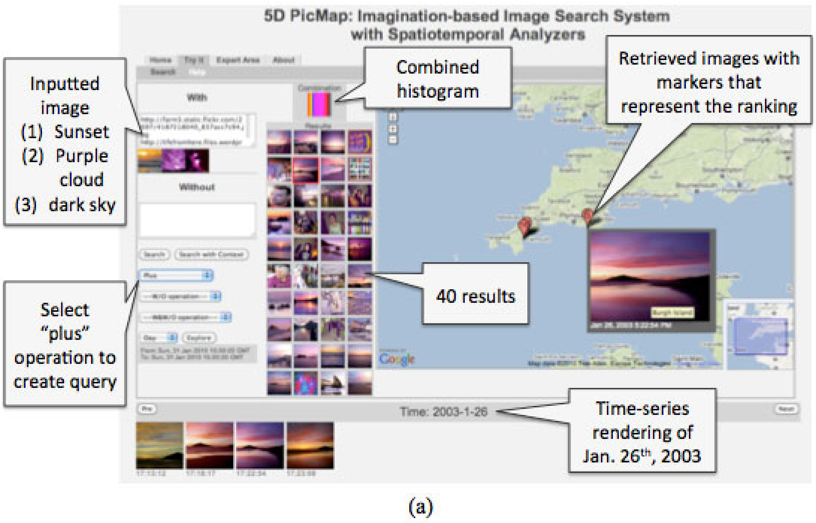

5D World Map
y
t
x
mapping
q1
doc1
doc2
docn
q2
qv
mapping
Multidimensional Vector Space For Semantic Calculation
Cultural Multimedia Databases

Fig. 4. 5D World Map System for world-wide viewing of Cultural and multimedia information resources

The 5D World Map system applied to cross-cultural multimedia computing visualizes world-wide and global relations among different areas and times in cultural aspects, by using dynamic mapping functions with temporal, spatial, semantic and impression-based computations.

Figure 5 (a) shows an image query processing environment with images for calculating correlations in colors between an impression-based query and images. Impression-based and highly-related images to the query are selected and allocated onto the global map with the spatial information for visualizing global aspects of spatially dependent features among different cultures.


5D PicMap: Imagination-based Image Search System with Spatiotemporal Analyzers
Inputted image (1) Sunset (2) Purple cloud (3) dark sky
Combined histogram
Retrieved images with markers that represent the ranking
Select "plus" operation to create query
40 results
Time: 2003-1-26
Time-series rendering of Jan. 26th, 2003
(a)

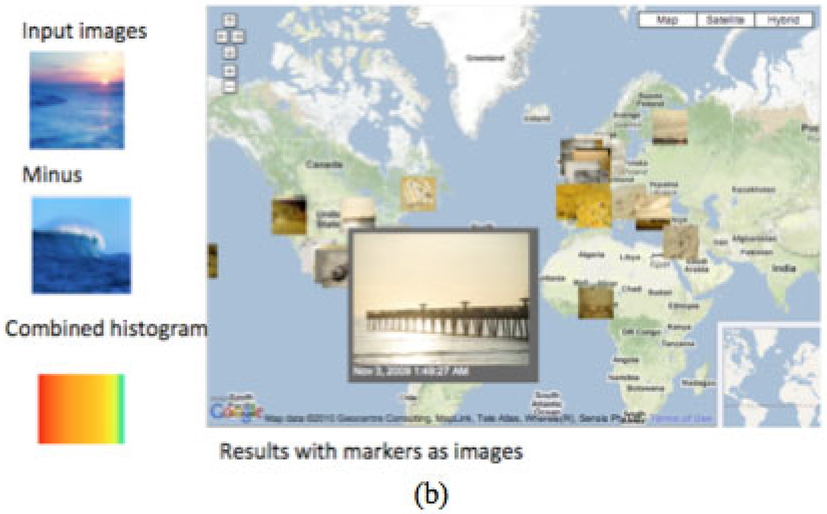


Fig. 5. 5D World Map System applied to visualization of world-wide and global relations among different areas and times in cultural aspects

Figure 5 (b) shows global comparisons among different areas in the context of color combinations. In this figure, the highly related images to the given context are mapped to the 5D world map as the thumbnail images, and various aspects among different areas can be globally visualized with some specific context.

5 Conclusion

The rapid progress of multimedia technology has realized the large scale of media data transfer and resource-accumulation in the world [11, 22, 23, 24]. Cross-cultural communication becomes important issues in global societies and communities connected in the world-wide scope. The innovative integration of large scale multimedia data management and cross-cultural computing will lead to new cross-cultural environments in our society. This paper has presented and summarized several new approaches to cross-cultural multimedia computing with multimedia system architectures and spatial, temporal, semantic and impression-based multimedia data analysis.

As our future work, we will extend our cross-cultural multimedia computing system to new international and collaborative research environments for realizing mutual understanding and knowledge sharing among different cultures.

Acknowledgments. We would like to express great thanks to Prof. Bernhard Thalheim (Christian-Albrechts-University Kiel) for his suggestive advices and our fruitful collaboration, continuing more than ten years, on the idea of cross-cultural computing among European and Asian countries.

References

[1] Brown, N.: Evolutionary models of music: From sexual selection to group to group selection. In: Perspectives in Ethology, pp. 231–281. Plenum Publishers, New York (2000)
[2] Cross, I.: Music, cognition, culture and evolution. Annals of the New York Academy of Sciences 930, 28–42 (2001)
[3] Goethe, W.J.: Theory of Colours, Trans. Charles Lock Eastlake, Cambridge. The M.I.T. Press, Massachusetts (1982)
[4] Harada, A. (ed.): Report of modeling the evaluation structure of KANSEI. Univ. of Tsukuba (1997)
[5] Hevner, K.: Expression in Music: A Discussion of Experimental Studies and Theories. Psychological Review 42, 186–204 (1935)
[6] Hevner, K.: Experimental Studies of the Elements of Expression in Music. American Journal of Psychology 48, 246–268 (1936)
[7] Hofstede, G.H., Hofstede, G.J.: Cultures and Organizations: Software of the Mind. McGraw-Hill Professional (2005)
[8] Ijichi, A., Kiyoki, Y.: A Kansei Metadata Generation Method for Music Data Dealing with Dramatic Interpretation. In: Information Modelling and Knowledge Bases, vol. XVI, pp. 170–182. IOS Press (May 2005)
[9] Imai, S., Kurabayashi, S., Kiyoki, Y.: A Music Retrieval System Supporting Intuitive Visualization by the Color Sense of Tonality. In: Proceedings of the 24th IASTED International Multi-Conference Databases and Applications (DBA 2006), pp. 153–159 (Feburary 2006)

[10] Kitagawa, T., Kiyoki, Y.: Fundamental Framework for Media Data Retrieval Systems using Media-lexico Transformation Operator in the case of musical MIDI data. In: Information Modeling and Knowledge Bases, vol. XII. IOS Press (2001)

[11] Kiyoki, Y., Kitagawa, T., Hayama, T.: A Metadatabase System for Semantic Image Search by a Mathematical Model of Meaning. ACM SIGMOD Record 23(4), 34–41 (1994)

[12] Kiyoki, Y., Kitagawa, T.: A Semantic Sssociative Search Method for Knowledge Acquisition. In: Information Modelling and Knowledge Bases, vol. VI, pp. 121–130. IOS Press (1995)

[13] Kiyoki, Y., Kitagawa, T., Hitomi, Y.: A Fundamental Framework for Realizing Semantic Interoperability in a Multidatabase Environment. International Journal of Integrated Computer-Aided Engineering (Special Issue on Multidatabase and Interoperable Systems) 2(1), 3–20 (1995)

[14] Kiyoki, Y., Kitagawa, T., Hayama, T.: A Metadatabase System for Semantic Image Search by a Mathematical Model of Meaning. In: Sheth, A., Klas, W. (eds.) Multimedia Data Management – using Metadata to Integrate and Apply Digital Media, ch. 7, McGrawHill (March 1998)

[15] Kiyoki, Y.: A Semantic Associative Search Method for WWW Information Resources. In: Proceedings of 1st International Conference on Web Information Systems Engineering (2000) (invited paper)

[16] Kiyoki, Y., Ishihara, S.: A Semantic Search Space Integration Method for Meta-level Knowledge Acquisition from Heterogeneous Databases. In: Information Modelling and Knowledge Bases, vol. XIV, pp. 86–103. IOS Press (May 2002)

[17] Kiyoki, Y., Chen, X.: A Semantic Associative Computation Method for Automatic Decorative Multimedia Creation with "Kansei" Information. In: The Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM 2009), 9 pages (January 2009) (invited paper)

[18] Kobayashi, S.: Color Image Scale (The Nippon Color & Design Research Institute ed., translated by Louella Matsunaga, Kodansha International, 1992)

[19] Le, M.H.: The role of Music in Second Language Learning: A Vietnamese Perspective. For Presentation at Combined 1999 Conference of the Australian Association for Research in Education and the New Zealand Association for Research in Education (1999)

[20] Nguyen, T.N.D., Sasaki, S., Kiyoki, Y.: 5D World PicMap: Imagination-based Image Search System with Spatiotemporal Analyzers. In: Proceedings of IADIS e-Society 2011 Conference, Avila, Spain, 8 pages (March 2011)

[21] Sasaki, S., Takahashi, Y, Y., Kiyoki, Y.: The 4D World Map System with Semantic and Spatiotemporal Analyzers. In: Information Modelling and Knowledge Bases, vol. XXI, 18 pages. IOS Press (2010)

[22] Sheth, A., Klas, W. (eds.): Multimedia Data Management - Using Metadata to Integrate and Apply Digital Media. MacGraw-Hill (March 1998)

[23] Thalheim, B.: Entity-relationship modeling – Foundations of database technology. Springer, Berlin (2000)

[24] Thalheim, B.: Application Development Based on Database Components. In: Information Modelling and Knowledge Bases, vol. 16, pp. 28–45. IOS Press (2004)

[25] Trang, N.N., Sasaki, S., Kiyoki, Y.: A Cross-Cultual Music Museum System With Impression-Based Analyzing Functions. In: Proceedings of IADIS e-Society 2011 Conference, Avila, Spain, 8 pages (March 2011)

[26] Dissanayake, E.: Antecedents of the temporal arts in early mother-infant interaction. In: The Origins of Music, pp. 389–410. MIT Press, Cambridge (2000)
[27] King, W.R.: A Research Agenda for the Relationships between Culture and Knowledge Management. Knowledge and Process Management 14, 226–236 (2007)
[28] Tosa, N., Matsuoka, S., Thomas, H.: Cultural Computing: SENetic Computer. In: ACM SIGGRAPH 2004 Emerging Technologies, Los Angeles, California, p. 11 (2004)
[29] Rauterberg, M.: From Personal to Cultural Computing: how to assess a cultural experience. In: uDayIV - Information Butzbar Machen, Pasbst Sience Publ. (2006)
[30] Nakatsu, R., Rauterberg, M., Salem, B.: Forms and theories of communication: from multimedia to Kansei Mediation. Multimedia Systems 11, 304–312 (2006)
[31] Heimbürger, A., Sasaki, S., Yoshida, N., Venäläinen, T., Linna, P., Welzer, T.: Cross-Cultural Collaborative Systems: Towards Cultural Computing. In: Information Modelling and Knowledge Bases, vol. XXI, pp. 403–417 (May 2010)
[32] Nguyen, N., Sasaki, S., Uraki, A., Kiyoki, Y.: A Semantic Metadata Extraction for Cross-cultural Music Environments by using Culture-based Music Samples and Filtering Functions. In: Proceedings of International Conference on Computer, Electrical, and Systems Science, and Engineering (ICCESSE 2010), May 26-28 (2010)
[33] Holsti, O., North, R.: Comparative Data from Content Analysis: Perception of History and Economic Variables in the 1914 Crisis. In: Merritt, R.L., Rokkan, S. (eds.) Comparing Nations: The Use of Quantitative Data in Cross-National Research, pp. 169–190 (1966)
[34] Zinnes, D.: A comparison of Hostile Behavior of Decision-Makers in Simulated historical Data. World Politics 18, 474–502 (1966)
[35] Batini, C., et al.: A Comparative analysis of methodologies for database schema integration. ACM Computing Surveys 18(4), 324–364 (1986)
[36] Egenhofer, M.J., et al.: Metric Details for Natural-Language Spatial Relations. ACM Trans. Information Systems 16(4), 295–321 (1998)
[37] Egenhofer, M.J.: Spatial SQL: A Query and Presentation Language. IEEE Transactions on Knowledge and Data Engineering 6(1), 86–95 (1994)
[38] Guting, R.H.: An Introduction to Spatial Database Systems. VLDB Journal 3, 357–399 (1994)
[39] Litwin, W.: An overview of the multidatabase system MRDSM. In: Proceedings of the 1985 ACM Annual Conference on The Range of Computing, pp. 524–533 (1985)
[40] Litwin, W., et al.: Interoperability of multiple autonomous databases. ACM Computing Surveys (CSUR) 22(3), 267–293 (1990)
[41] Yoshida, N., Kiyoki, Y.: An associative search method based on symbolic filtering and semantic ordering. In: Proceedings of the 7th IFIP 2.6 Working Conference on Database Semantics, pp. 215–237 (1997)
[42] Hosokawa, Y., Kiyoki, Y.: Functional and parallel query processing and query optimization for multidatabase systems. In: Proc. the 17 th IASTED International Conference on Applied Informatics, pp. 101–106 (1999)
[43] Ishibashi, N., Kiyoki, Y.: Meta-Chronicle: A Spatial and Temporal Multidatabase System and its Application to Histories. In: Proceedings of IEEE International Symposium on Applications and the Internet (SAINT 2004) - the International Workshop on Cyberspace Technologies and Societies (IWCTS 2004), pp. 515–522 (2004)

Author Index