

Automated Reasoning on Conceptual Schemas

Edited by

Diego Calvanese¹, Sven Hartmann², and Ernest Teniente³

1 Free University Bozen-Bolzano, IT, calvanese@inf.unibz.it

2 TU Clausthal, DE, sven.hartmann@tu-clausthal.de

3 UPC – Barcelona, ES, teniente@essi.upc.edu

Abstract

This report documents the outcomes of the Dagstuhl Seminar 13211 “Automated Reasoning on Conceptual Schemas”. The quality of an information system is largely determined early in the development cycle, i.e., during requirements specification and conceptual modeling since errors introduced at these stages are usually much more expensive to correct than errors made during design or implementation. Thus, it is desirable to prevent, detect, and correct errors as early as possible in the development process by assessing the correctness of the conceptual schemas built. The high expressivity of conceptual schemas requires to adopt automated reasoning techniques to support the designer in this important task.

Research in this area can be classified according to two different dimensions. On the one hand, according to the language used to specify the conceptual schema. On the other hand, according to whether reasoning is performed on the structural schema alone, or also on its dynamic aspects. We find interesting and promising results from all these communities which have usually worked isolatedly. Therefore, the aim of this seminar was to allow them to communicate with each other to avoid duplicate effort and to exploit synergies. The research questions that were pursued in the seminar included, among others: (i) Does it make sense to renounce to decidability to be able to handle the full expressive power of the language used with and without textual integrity constraints? (ii) Which is the current state of the achievements as far as reasoning on the behavioral part is concerned? (iii) Are the existing techniques and tools ready to be used in an industrial environment? (iv) Which are the new challenges for automated reasoning on conceptual schemas?

Seminar 19–24 May, 2013 – www.dagstuhl.de/13211

1998 ACM Subject Classification D.2.4 [Software Engineering] Software/Program Verification, F.3.1 [Logics and meanings of programs] Specifying, Verifying and Reasoning about Programs

Keywords and phrases Automated Reasoning, Conceptual Schema of an Information System, Validation, Verification

Digital Object Identifier 10.4230/DagRep.3.5.43

Edited in cooperation with Xavier Oriol

1 Executive Summary

Diego Calvanese

Sven Hartmann

Ernest Teniente

License  Creative Commons BY 3.0 Unported license
© Diego Calvanese, Sven Hartmann, and Ernest Teniente

This Dagstuhl Seminar brought together 37 researchers from 16 countries across disciplines related to automated reasoning on conceptual schemas. The participants’ expertise covered



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Automated Reasoning on Conceptual Schemas, *Dagstuhl Reports*, Vol. 3, Issue 5, pp. 43–77

Editors: Diego Calvanese, Sven Hartmann, and Ernest Teniente



DAGSTUHL
REPORTS

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

	Monday	Tuesday	Wednesday	Thursday	Friday
08:30		Reasoning about the Dynamics			
09:00	<i>Opening</i> Reasoning on Structural Schemas (I)		Break Out Session	Break Out Session	Group Conclusions
10:00	Coffee Break				
10:30	Reasoning on Structural Schemas (II)	New Challenges	Break Out Session	Break Out Session	Group Conclusions
12:00	Lunch				
14:00	Reasoning on Structural Schemas (III)	Reasoning about Mappings	<i>Excursion</i>	Reports of the Break Out Sessions	
15:30	Coffee Break			Coffee Break	
16:00	Extensions	Reasoning about Dependencies		Discussion	
18:00	Dinner				

■ **Figure 1** Timetable of the seminar.

the three most popular languages used to specify the conceptual schema, i.e., Entity-Relationship (ER), Unified Modeling Language (UML) and Object-Role Modeling (ORM); either addressing reasoning only on the static (i.e., structural) schema alone or reasoning also on the elements of a conceptual schema that capture the dynamic (i.e., behavioral) aspects of a system.

Monday and Tuesday were devoted to short presentations from the participants of their most recent achievements in the field.

On Wednesday and Thursday morning the participants were allocated to three different groups, in parallel break out sessions, each one of them addressing a different aspect related to the topic of the workshop:

- On the practical applicability of current techniques for reasoning on the structural schema;
- Reasoning about the conceptual schema components capturing dynamic aspects;
- New challenges for automated reasoning on conceptual schemas.

The organizers asked each group to share the experiences of their participants and to try to identify the most pressing and challenging research issues or open problems for the aspect it addressed. Each group presented a summary of their results on Thursday afternoon. Thursday evening and Friday morning were devoted to a discussion about the outcomes of each group aiming at trying to come up with a roadmap for automated reasoning on conceptual schemas, something which was shown to be harder than expected.

The concrete timetable of the seminar is shown in Figure 1.

2 Table of Contents

Executive Summary

<i>Diego Calvanese, Sven Hartmann, and Ernest Teniente</i>	43
--	----

Overview of Talks

Reasoning over Conceptual Data Models: The Description Logic Approach <i>Alessandro Artale</i>	48
UML class diagrams – decision, identification and repair of correctness and quality problems <i>Mira Balaban</i>	49
A Problem Statement: Representation of Instance-Derivations Based on Dependencies <i>Joachim Biskup</i>	50
Employing Automatic Reasoning on Conceptual Schemas <i>Joachim Biskup</i>	50
Incremental inconsistencies detection with low memory overhead <i>Xavier Blanc</i>	51
Modeling@SAP: Why Class Models Are Rarely Used <i>Achim D. Brucker</i>	51
Reasoning over Secure Business Processes <i>Achim D. Brucker</i>	52
Modeling and Reasoning over Processes and Data <i>Diego Calvanese</i>	52
Preliminary Report on an Algebra of Lightweight Ontologies <i>Marco A. Casanova</i>	53
OCL2FOL: Using SMT solvers to automatically reason on conceptual schemata with OCL constraints <i>Carolina Dania</i>	53
Metrics for visual notations <i>Sophie Dupuy-Chessa</i>	54
“Automating Reasoning on Conceptual Schemas” in FamilySearch TM – a Large-Scale Reasoning Application <i>David W. Embley</i>	54
Constraints on Class Diagrams <i>Ingo Feinerer</i>	54
ORM2: formalisation and encoding in OWL2 <i>Enrico Franconi</i>	55
Reasoning over Order Dependencies for Relational Schema <i>Parke Godfrey</i>	55
Exploring UML and OCL Model Properties with Relational Logic <i>Martin Gogolla</i>	56

Armstrong Instances as a Reasoning Aid <i>Sven Hartmann</i>	56
Automated Design of Updateable Database Views: a Framework for Possible Strategies <i>Stephen J. Hegner</i>	57
An ontology-driven unifying metamodel for UML Class Diagrams, EER, and ORM2 <i>C. Maria Keet</i>	57
Temporal Extended Conceptual Models <i>Roman Kontchakov</i>	58
ProB: Solving Constraints on Large Data and Higher-Order Formal Models <i>Michael Leuschel</i>	59
A Declarative Approach to Distributed Computing <i>Jorge Lobo</i>	59
Reasoning on conceptual schemas of spatial data <i>Stephan Maes</i>	60
On BDD, Finite controllability and the BDD/FC conjecture <i>Jerzy Marcinkowski</i>	61
On the Relationship Between OBDA and Relational Mapping <i>Marco Montali</i>	61
Reasoning about the Effect of Structural Events in UML Conceptual Schemas <i>Xavier Oriol</i>	61
Information and dependency preserving BCNF decomposition algorithm via attribute splitting <i>Elena V. Ravve</i>	62
Semantic-based Mappings <i>Guillem Rull</i>	63
The Curse of Restructuring in Dependency Theory <i>Klaus-Dieter Schewe</i>	63
AuRUS: Automated Reasoning on UML Schemas <i>Ernest Teniente</i>	64
Visual reasoning with (functional) dependencies <i>Bernhard Thalheim</i>	64
Validation of Complex Domain-Specific Modeling Languages <i>Dániel Varró</i>	65
Reasoning about Dependencies in Schema Mappings <i>Qing Wang</i>	65

Working Groups

On the Practical Applicability of Current Techniques for Reasoning on the Structural Schema <i>Ernest Teniente</i>	66
Reasoning about the Conceptual Schema Components Capturing Dynamic Aspects <i>Diego Clavanes</i>	69

New Challenges for Automated Reasoning on Conceptual Schemas <i>Sven Hartmann</i>	71
Seminar program	75
Participants	77

3 Overview of Talks

3.1 Reasoning over Conceptual Data Models: The Description Logic Approach

Alessandro Artale (Free University of Bozen-Bolzano, IT)

License © Creative Commons BY 3.0 Unported license
© Alessandro Artale

Joint work of Artale, Alessandro; Calvanese, Diego; Kontchakov, Roman; Ryzhikov, Vladislav; Zakharyashev, Michael;

Main reference A. Artale, D. Calvanese, R. Kontchakov, V. Ryzhikov, M. Zakharyashev, “Reasoning over extended ER models,” in: *Conceptual Modeling – ER 2007*, LNCS, Vol. 4801, pp. 277–292, Springer, 2007.

URL http://dx.doi.org/10.1007/978-3-540-75563-0_20

In this talk we show how reasoning techniques developed in the field of knowledge representation (Description Logics, in particular) can be used to check quality properties of Conceptual Models [4], i.e., schema consistency, class consistency, instance checking and model checking [6, 5, 3, 1]. We consider various fragments of the language of Extended Entity-Relationship (EER) diagrams, which includes a number of constructs: ISA between entities and relationships, disjointness and covering of entities and relationships, cardinality constraints for entities in relationships and their refinements as well as multiplicity constraints for attributes.

The main results are obtained by mapping ER constructs to, so called, DL-Lite [7, 2] logics, thus showing the usefulness of such languages for reasoning over conceptual models and ontologies. The talk will also emphasise the difference between the database and the ontology point of view of Conceptual Models.

References

- 1 A. Artale, D. Calvanese, R. Kontchakov, V. Ryzhikov, and M. Zakharyashev. Reasoning over extended ER models. In *Proc. of the 26th International Conference on Conceptual Modeling (ER’07)*, volume 4801, Auckland, New Zealand, Nov. 2007. Lecture Notes in CS, Springer.
- 2 A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyashev. The DL-Lite family and relations. *Journal of Artificial Intelligence Research (JAIR)*, 36:1–69, 2009.
- 3 A. Artale, D. Calvanese, and A. Ibáñez-García. Full satisfiability of uml class diagrams. In *Proc. of the 29th International Conference on Conceptual Modeling (ER-10)*, 2010.
- 4 C. Batini, S. Ceri, and S.B. Navathe. *Conceptual Database Design, an Entity-Relationship Approach*. Benjamin and Cummings Publ. Co., 1992.
- 5 D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1–2):70–118, 2005.
- 6 D. Calvanese, M. Lenzerini, and D. Nardi. Unifying class-based representation formalisms. *J. of Artificial Intelligence Research*, 11:199–240, 1999.
- 7 D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 260–270, 2006.

3.2 UML class diagrams – decision, identification and repair of correctness and quality problems

Mira Balaban (Ben Gurion University – Beer Sheva, IL)

License © Creative Commons BY 3.0 Unported license
© Mira Balaban

Joint work of Balaban, Mira; Maraee Azzam; Sturm Arnon; Kifer Michael; Khitron Igal

Main reference M. Balaban, A. Maraee, “Finite satisfiability of UML class diagrams with constrained class hierarchy,” *ACM Trans. on Software Engineering and Methodology (TOSEM)*, 22(3), 24pp., ACM, 2013.

URL <http://dx.doi.org/10.1145/2491509.2491518>

We have developed efficient methods for analysis of correctness and quality problems in UML class diagrams. The ultimate goal is to have a rich support for static analysis of models, so to enable the development of advanced model level IDEs. Our methods are implemented in our FiniteSatUSE tool and its associated catalog of correctness anti-patterns. The methods:

1. Detection of Finite Satisfiability problems.
2. Identification of the cause for a Finite Satisfiability or a Consistency problem.
3. Repair – for typical problems. Based on the catalog.
4. Simplification of diagrams – remove redundant constraints.
5. Completion of diagrams – discover hidden constraints.

Most methods apply to subsets of UML class diagrams – depending on included constraints and structure of the diagram. Some methods are complete for certain subsets, but become incomplete when new constraints are added.

Various questions arise from this research:

1. The need for abstraction means in class diagrams – Our Pattern Class Diagram language is a start.
2. Efficient discovery of patterns in a class diagram.
3. Creation of a benchmark of class diagrams, for evaluation of analysis algorithms – Metric directed class diagram constructor; e.g., construct a class diagram that satisfies a structure metric of presence of association- cycles, or a size-metric of proportion of classes and associations.
4. Integration of (1) a UML tool; with (2) a reasoning underlying module (our F-OML project), and with (3) the above methods.


All relevant material is in: <http://www.cs.bgu.ac.il/modeling/>

References

- 1 M. Balaban and A. Maraee *Finite Satisfiability of UML Class Diagrams with Constrained Class Hierarchy*. TOSEM, to appear. ACM (2013).
- 2 M. Balaban and A. Maraee *Inter-association Constraints in UML2: Comparative Analysis, Usage Recommendations, and Modeling Guidelines*. MoDELS (2012).
- 3 M. Balaban and M. Kifer *Logic-based Model-Level Software Development with F-OML*. MoDELS (2011).
- 4 M. Balaban and A. Maraee and A. Sturm *A Pattern-Based Approach for Improving Model Quality*. submitted (2013).

3.3 A Problem Statement: Representation of Instance-Derivations Based on Dependencies

Joachim Biskup (TU Dortmund, DE)

License  Creative Commons BY 3.0 Unported license
© Joachim Biskup

A relational database schema is specified by a set of relation names with attributes and a set of dependencies that constrain the possible instance relations. Regarding the dependencies, mostly two kinds of derivations have been investigated, namely deciding on implications and verifying satisfaction.

In this talk, we will study another kind of derivations, namely: whether and how a user who has an incomplete view on the instance relations can derive further nontrivial details of the instance relations based on the dependencies. More specifically, we would like to obtain a concise representation of all options for such derivations. Our interest is motivated by applications in the field of inference control; it might also be relevant for avoiding redundancy when designing a database.

Intuitively, given the dependencies we would like find a collection of “derivation-skeletons” such that the following properties are achieved: correctness for all instances, completeness for all instances, and neither local nor global redundancy.

Having no substantial results so far, we basically only have the following conjecture: A set of dependencies (having some nice properties like being “full” (not embedded)), admits a finite collection of “derivation-skeletons” with the required properties iff the given set is some sense “acyclic” (still to be made precise).

3.4 Employing Automatic Reasoning on Conceptual Schemas

Joachim Biskup (TU Dortmund, DE)

License  Creative Commons BY 3.0 Unported license
© Joachim Biskup

Joint work of Biskup, Joachim; Hartmann, Sven

We suggest the following classification of how automatic reasoning on conceptual schemas is employed:

1. At design time:
 - prove assurances for given schema:
 - * assurances for a single state:
 - ...
 - * assurances for sequences of states:
 - ...
 - transform given schema into “better” one (satisfying more useful assurances)
 - exhibit inherent conflicts/tradeoffs to achieve useful assurances
 - integrate new aspects or independently specified aspects
 - generate “closed” views from specification of informational needs
 - generate “representative” instances for illustrating practical consequences
 - estimate future costs (runtime, space, communication, ...)
 - 2. At run time, continued employment using the assurances proven at design time:
 - improve quality of answers to queries

- supply/suggest missing inputs for updates of data (instances)
- ensure meaningful results of complex data manipulations
- support prediction and optimization of operational requests
- support schema consolidation
- assist in enforcing assurances during evolvement of informational needs
- enable interoperation/communication with other agents, in particular for complex querying

3.5 Incremental inconsistencies detection with low memory overhead

Xavier Blanc (University of Bordeaux, FR)

License © Creative Commons BY 3.0 Unported license
© Xavier Blanc

As ensuring models' consistency is a key concern when using a model-based development approach, model inconsistency detection has received significant attention over the last years. To be useful, inconsistency detection has to be sound, efficient and scalable. Sound means that inconsistencies detected by a tool should be correct, and not mislead the developers in their design tasks. Efficient means that detection has to be executed as fast as possible because developers always want to know the impact of their modification immediately after having performing them. Scalability is however a little more complex as a concern and usually defines specific requirements that are deemed important depending on the context. In this talk, we presented a new incremental inconsistency detection approach that emphasis on scalability as it only consumes a small and model size-independent amount of memory.

3.6 Modeling@SAP: Why Class Models Are Rarely Used

Achim D. Brucker (SAP Research – Karlsruhe, DE)

License © Creative Commons BY 3.0 Unported license
© Achim D. Brucker

In 1999, SAP started to combine the Unified Modeling Language (UML) and the Fundamental Modeling Concepts (FMC) language (for details, see <http://www.fmc-modeling.org/fmc-and-tam/>). The result is an SAP internal standard for modeling, called Technical Architecture Modeling (TAM). TAM comprises block diagrams, component diagrams, package diagrams, class diagrams, activity diagrams, sequence diagrams, state diagrams, and use case diagrams. TAM is used for both conceptual modeling as well as design modeling.

While many works on reasoning on conceptual schemas focus on class diagrams and state diagrams, the most often used diagram type used for conceptual modeling at SAP is the block diagram. For example, class models are used rarely, as they are “too close to real code.” In general, developers and architects prefer high-level structural diagrams (e.g., block diagrams), thus we need to ask ourselves the questions, if we can reason over such models and what kind of properties help to improve the software development.

3.7 Reasoning over Secure Business Processes

Achim D. Brucker (SAP Research – Karlsruhe, DE)

License  Creative Commons BY 3.0 Unported license
© Achim D. Brucker

Enterprise systems are often process-driven. In such systems, high-level process-models play an important role both for communicating business requirements between domain experts and system experts as well as for system implementation. Since several years, enterprise systems need to fulfill an increasing number of the security and compliance requirements. Thus, there is an increasing demand for integrating high-level security and compliance requirements into process models.

In general, we present an approach for reasoning over secure process-models, i.e., process-models containing high-level security and compliance requirements. In particular, this approach helps to detect non-compliant or inconsistent process-models early during both the modeling of business processes as well as during system development.

References

- 1 A.D. Brucker and I. Hang. Secure and compliant implementation of business process-driven systems. In Marcello La Rosa and Pnina Soffer, editors, *Joint Workshop on Security in Business Processes (SBP)*, volume 132 of *Lecture Notes in Business Information Processing (LNBIP)*, pages 662–674. Springer-Verlag, 2012.
- 2 A.D. Brucker, I. Hang, G. Lückemeyer, and R. Ruparel. SecureBPMN: Modeling and enforcing access control requirements in business processes. In *ACM symposium on access control models and technologies (SACMAT)*, pages 123–126. ACM Press, 2012.
- 3 L. Compagna, P. Guilleminot, and A.D. Brucker. Business process compliance via security validation as a service. In Manuel Oriol and John Penix, editors, *Testing Tools Track of International Conference on Software Testing, Verification, and Validation (Tools@ICST)*. IEEE Computer Society, 2013.

3.8 Modeling and Reasoning over Processes and Data

Diego Calvanese (Free University of Bozen-Bolzano, IT)

License  Creative Commons BY 3.0 Unported license
© Diego Calvanese

Joint work of Calvanese, Diego; Bagheri Hariri, Babak; Montali, Marco; Santoso, Ario; De Giacomo, Giuseppe; Deutsch, Alin

Main reference B. Bagheri Hariri, D. Calvanese, G. De Giacomo, A. Deutsch, M. Montali, “Verification of relational data-centric dynamic systems with external services,” arXiv:1203.0024v1 [cs.DB], 2012.

URL <http://arxiv.org/abs/1203.0024v1>

Data and processes are just two sides of the same coin, and for several activities related to the analysis and design of systems it is therefore important to capture both static and dynamic aspects in a uniform way. Data-centric dynamic systems (DCDSs) are systems where both the process controlling the dynamics and the manipulation of data are equally central, and are captured in a pristine way, abstracting from specific features of concrete models. DCDSs allow one to capture commonly adopted models for data and processes, such as artifact systems modeled as finite state machines or through the Guard-Stage- Milestone (GSM) model. Hence, they are well suited for the formal analysis and verification of temporal properties over the evolution of such systems. In the talk we present DCDSs and discuss the results on decidability of verification of first-order variants of μ -calculus over such

systems. We also point to extensions of DCDSs with a semantic level, allowing one to capture properties at a higher level of abstraction, and taking into account semantic constraints during verification.

3.9 Preliminary Report on an Algebra of Lightweight Ontologies

Marco A. Casanova (PUC – Rio de Janeiro, BR)

License © Creative Commons BY 3.0 Unported license
© Marco A. Casanova

Joint work of Casanova, Marco A.; Sacramento, Eveline R.; Macêdo, José A. F.; Pinheiro, Ângela M. A.; Vidal, Vânia M. P.; Breitman, Karin K.; Furtado, Antonio L.

Main reference M.A. Casanova, J.A.F. de Macêdo, E.R. Sacramento, Â.M.A. Pinheiro, V.M.P. Vidal, K.K. Breitman, A.L. Furtado, “Operations over lightweight ontologies,” in: *On the Move to Meaningful Internet Systems (OTM’12)*, LNCS, Vol. 7566, pp. 646–663, Springer, 2012.

URL http://dx.doi.org/10.1007/978-3-642-33615-7_14

We argue that certain familiar ontology design problems are profitably addressed by treating ontologies as theories and by defining a set of operations that create new ontologies, including their constraints, out of other ontologies. Such operations extend the idea of namespaces to take into account constraints.

Consider first the problem of designing an ontology to publish data on the Web. If the designer follows the Linked Data principles, he must select known ontologies, as much as possible, to organize the data so that applications can dereference the URIs that identify vocabulary terms in order to find their definition. We argue that the designer should go further and analyze the constraints of the ontologies from which he is drawing the terms to construct his vocabulary. Furthermore, he should publish the data in such a way that the original semantics of the terms is preserved. To facilitate ontology design from this perspective, we introduce three operations on ontologies, called projection, union and deprecation.

Consider now the problem of comparing the expressive power of two ontologies, $O = (V, \Sigma)$ and $O' = (V', \Sigma')$. If the designer wants to know what they have in common, he should create a mapping between their vocabularies and detect which constraints hold in both ontologies, after the terms are appropriately mapped. The intersection operation answers this question. We argued elsewhere (Casanova et al., 2010) that intersection is also useful to address the design of mediated schemas that combine several export schemas in a way that the data exposed by the mediator is always consistent.

On the other hand, if the designer wants to know what holds in $O = (V, \Sigma)$, but not in $O' = (V', \Sigma')$, he should again create a mapping between their vocabularies and detect which constraints hold in the theory of Σ , but not in the theory of Σ' , after the terms are appropriately mapped. The difference operation answers this question.

3.10 OCL2FOL: Using SMT solvers to automatically reason on conceptual schemata with OCL constraints

Carolina Dania (IMDEA Software Institute, ES)


License © Creative Commons BY 3.0 Unported license
© Carolina Dania

In this talk we present a mapping from a rich subset of OCL expressions into first order logic, and discuss how this mapping can be used to automatically check, using SMT solvers, the

satisfiability of conceptual schemata with OCL constraints. We also present our OCL2FOL tool, which implements our mapping and generates, for each conceptual schema and OCL constraints, the corresponding satisfiability problem for Z3 or Yices.

3.11 Metrics for visual notations

Sophie Dupuy-Chessa (LIG – Grenoble, FR)


License  Creative Commons BY 3.0 Unported license
© Sophie Dupuy-Chessa

Joint work of Le Pallec, Xavier; Mandran, Nadine; Genon Nicolas

The maturity of Model Driven Engineering facilitates the development of domain specific languages. The creation of the languages relies on the definition of metamodels, but also on their corresponding visual notations. But one can wonder what is the quality of a new language. It can result in inunderstandable diagrams with inappropriate notations. Then our goal is to evaluate the quality of metamodels and notations by metrics, which are integrated in a (meta)modeling environment. In particular, we are studying metrics for visual notations in order to automate the measure of quality for a notation and its derived diagrams.

3.12 “Automating Reasoning on Conceptual Schemas” in FamilySearchTM – a Large-Scale Reasoning Application


David W. Embley (Brigham Young University, US)

License  Creative Commons BY 3.0 Unported license
© David W. Embley

Among its many other projects, FamilySearch.org has scanned and OCREd 85,000 books filled with family-history information—an estimated 4.25×10^{12} “facts” of interest. We wish to extract and organize these facts for semantic search with respect to a conceptual model. Reasoning applies in several ways: (1) traditional satisfiability checks over potentially more expressive constraints, (2) inference with large sets of assertions—about half of the facts are inferred; (3) constraint violation search—unlike many database applications, it’s almost certain that the facts will not satisfy the constraints of the conceptual model, so finding the myriad of discrepancies is of interest; and (4) uncertainty—both facts and constraints are uncertain, so probabilistic description logics are of interest. The presentation will likely have more questions than answers, and in that sense, could lead to interesting discussion and possible directions for future work.

3.13 Constraints on Class Diagrams

Ingo Feinerer (TU Wien, AT)

License  Creative Commons BY 3.0 Unported license
© Ingo Feinerer

We give a short overview of our approach for encoding UML class diagrams for checking satisfiability and for computing minimal instances. We aim for efficient and lightweight

techniques, like integer linear programming and flow networks, that scale well also in industrial-scale application domains. As open problems for discussion we highlight a few constraints, typically expressed via OCL, that we found relevant in industrial settings but which are still challenging from a research perspective, like association chains or equations over them.

3.14 ORM2: formalisation and encoding in OWL2

Enrico Franconi (Free University of Bozen-Bolzano, IT)

License © Creative Commons BY 3.0 Unported license
© Enrico Franconi

Joint work of Franconi, Enrico; Mosca, Alessandro

Main reference E. Franconi, A. Mosca, D. Solomakhin, “ORM2: Formalisation and encoding in OWL2,” in: On the Move to Meaningful Internet Systems (OTM’12), LNCS, Vol. 7567, pp. 368–378, Springer, 2012.

URL http://dx.doi.org/10.1007/978-3-642-33618-8_51

The introduction of a provably correct encoding of a fragment of ORM2 (called ORM2zero) into a decidable fragment of OWL2, opened the doors for the definition of dedicated reasoning technologies supporting the quality of the schemas design.

3.15 Reasoning over Order Dependencies for Relational Schema

Parke Godfrey (York University – Toronto, CA)

License © Creative Commons BY 3.0 Unported license
© Parke Godfrey

Dependency theory has played important roles for relational databases, with functional dependencies (FDs) as a hallmark. Structural constraints provide a framework for formalizing good design, as by normalization and decomposition, and for reasoning over schemas. Much has been studied on how to reason effectively over dependencies. Armstrong’s Axioms provide a sound and complete axiomatization for FDs, which led to insights for efficient inference procedures. Such “reasoners” are easier to understand and prove much more efficient than general reasoning from first principles.

Dependencies have also played a critical role on the physical side of database systems. They are used within query optimization extensively. For instance, FDs can be used to convert one group-by specification as stated within an SQL query into a logically equivalent one, but one more amenable to an efficient query plan. Thus, query optimizers incorporate limited reasoners for dependencies.

Order dependencies (ODs) generalize functional dependencies. ODs are to FDs as order by is to group by. An OD tells that when data is sorted with respect to one specification, it then must also be sorted with respect to another. We have been studying ODs in depth recently, and we know much more about them now. We have devised a sound and complete axiomatization for them as Armstrong’s Axioms are for FDs. From this, we are developing efficient inference procedures for them. We know much about the complexities of inference tasks for ODs. We have shown a proper hierarchy for OD classes. Our motivation for this endeavor has been, in large part, to use ODs effectively within query optimization. ODs offer a powerful means to exchange one interesting order on a data stream (within the runtime of

the query plan) for another (when it is known the second implies the first), which broadens the space of possible plans.

However, ODs could also play a vital role in design, as do FDs. We do not understand yet how ODs can be viewed as structural constraints, and what roles they might play in schemas and reasoning over schemas. This is a worthwhile topic to explore.

3.16 Exploring UML and OCL Model Properties with Relational Logic

Martin Gogolla (Universität Bremen, DE)

License © Creative Commons BY 3.0 Unported license

© Martin Gogolla

Joint work of USE Development Team

Main reference M. Kuhlmann, M. Gogolla, “From UML and OCL to relational logic and back,” in: *Model Driven Engineering Languages and Systems, LNCS, Vol. 7590*, pp. 415–431, Springer, 2012.

URL http://dx.doi.org/10.1007/978-3-642-33666-9_27

URL <http://sourceforge.net/apps/mediawiki/useocl/index.php>

Modeling Languages like UML or EMF employ OCL in order to precisely model systems in terms of a conceptual schema for applications. Complex UML and OCL models therefore represent a crucial part of model-based engineering, as they allow to formally describe central system properties like model consistency or constraint independence, among other important properties. We discuss a lightweight model validation and verification method based on efficient SAT solving techniques. Our work relies on a transformation from UML class diagrams and OCL concepts into relational logic. Relational logic in turn represents the source for advanced SAT-based model finders. The approach allows us to explicitly benefit from the efficient handling of relational logic and to interpret found results backwards in terms of UML and OCL. We explain our ideas with an example from the original work by Peter Chen on conceptual modeling with the entity-relationship model.

References

- 1 M. Kuhlmann and M. Gogolla. From UML and OCL to Relational Logic and Back. In Robert France, Juergen Kazmeier, Ruth Breu, and Colin Atkinson, editors, *Proc. 15th Int. Conf. Model Driven Engineering Languages and Systems (MoDELS'2012)*, pages 415–431. Springer, Berlin, LNCS 7590, 2012.

3.17 Armstrong Instances as a Reasoning Aid

Sven Hartmann (TU Clausthal, DE)

License © Creative Commons BY 3.0 Unported license

© Sven Hartmann

Joint work of Hartmann, Sven; Link, Sebastian; Ferrarotti, Flavio; Köhler, Henning; Le, Van; Leck, Uwe; Thalheim, Bernhard; Trinh, Thu

In our talk we survey recent results on the structure, existence and computation of Armstrong instances for general cardinality constraints and functional dependencies in partial databases. Leading database design tools recommend the creation of data samples to validate, communicate, and evolve the database models they produce. Armstrong instances are perfect data samples in the sense that they satisfy exactly those integrity constraints within a particular constraint class that are logical consequences of the specified constraint set. They can be queried by the data engineer to test the implication of constraints within the constraint class

of interest. We discuss characterizations and constructions of Armstrong instances, and list open problems that we suggest for future research.

3.18 Automated Design of Updateable Database Views: a Framework for Possible Strategies

Stephen J. Hegner (University of Umeå, SE)

License © Creative Commons BY 3.0 Unported license
© Stephen J. Hegner

There has been substantial amount of work on the subject of how updates to database views should be supported. However, there has been little reported on how to design views which meet certain requirements for information content and furthermore support a certain set U of updates. In this work, some ideas on how to automate the design of views which are updateable via the constant-meet-complement strategy are presented. The design process itself may require some flexibility in the choice of the view to be updated, since a larger view will generally admit a larger set of updates. Thus, the process consists of the identification of a pair of views (Γ, Γ') , in which Γ is the view to be updated and Γ' is a meet complement which supports all updates in U . The constraint on Γ is that it recapture a certain prespecified minimal amount of information I_{\min} from the main schema, but not more than a prespecified upper bound I_{\max} . Since meet complements are precisely those which admit an embedded cover of the dependencies of the main schema, the key to an effective realization of this approach is to find methods for the efficient identification of embedded covers.

3.19 An ontology-driven unifying metamodel for UML Class Diagrams, EER, and ORM2

C. Maria Keet (University of KwaZulu-Natal – Durban, ZA)

License © Creative Commons BY 3.0 Unported license
© C. Maria Keet
Joint work of Keet, C. Maria; Fillottrani, Pablo Rubén


Software interoperability may be achieved by using their respective conceptual data models. However, each model may be represented in a different conceptual data modelling language for the tool's purpose or due to legacy issues. Several translations between small subsets of language features are known, but no unified model exists that includes all their language features. Aiming toward filling this gap, we designed a common and unified, ontology-driven, metamodel covering and unifying EER, UML Class Diagrams v2.4.1, and ORM2. We present the static, structural, components of the metamodel, highlighting the common entities and summarizing some modelling motivations. This metamodel will be taken as the basis for a comprehensive formalization, which afterward may be used for reasoning over the structural components of individual and linked conceptual models represented in any of UML, EER, or ORM2. More details of the project and the related papers are available online at <http://www.meteck.org/SAAR.html>

References

- 1 C.M. Keet, P.R. Fillottrani. Toward an ontology-driven unifying metamodel for UML Class Diagrams, EER, and ORM2. *32nd International Conference on Conceptual Modeling (ER'13)*. 11-13 November, 2013, Hong Kong. Springer LNCS (accepted).
- 2 C.M. Keet, P.R. Fillottrani. Structural entities of an ontology-driven unifying metamodel for UML, EER, and ORM2. *3rd International Conference on Model and Data Engineering (MEDI'13)*. September 25-27, 2013, Amantea, Calabria, Italy. Springer LNCS (accepted).

3.20 Temporal Extended Conceptual Models

Roman Kontchakov (Birkbeck College London, GB)

License  Creative Commons BY 3.0 Unported license
© Roman Kontchakov

In this talk we discuss the temporal description logics [3], designed for reasoning about temporal conceptual data models, and investigate their computational complexity. Our formalisms are based on the DL-Lite family of description logics with three types of concept inclusions (ranging from atomic concept inclusions and disjointness to the full Booleans), as well as cardinality constraints and role inclusions [4]. In the temporal dimension, they capture future and past temporal operators on concepts, flexible and rigid roles, the operators ‘always’ and ‘some time’ on roles, data assertions for particular moments of time and global concept inclusions. The logics are interpreted over the Cartesian products of object domains and the flow of time $(Z, <)$, satisfying the constant domain assumption. We prove [2] that the most expressive of our temporal description logics (which can capture lifespan cardinalities and either qualitative or quantitative evolution constraints) turn out to be undecidable. However, by omitting some of the temporal operators on concepts/roles or by restricting the form of concept inclusions we obtain logics whose complexity ranges between PSpace and NLogSpace. These positive and encouraging results were obtained by reduction to various clausal fragments of propositional temporal logic, which opens a way to employ propositional or first-order temporal provers for reasoning about temporal data models.

In the second part of the talk we focus on ontology-based data access (OBDA), where the most important technique for answering queries is to rewrite the given ontology and the query into a single first-order query that can be evaluated directly over the data. The current W3C standard for OBDA is OWL 2 QL which is based on the DL-Lite family of description logics. Our aim is to extend standard atemporal OBDA to data with validity time and ontologies that are suitable for temporal conceptual modelling. To this end, we design a temporal description logic, TQL, that extends the standard ontology language OWL 2 QL, provides basic means for temporal conceptual modelling and ensures first-order rewritability of conjunctive queries for suitably defined data instances with validity time [1].

References

- 1 A. Artale, R. Kontchakov, F. Wolter and M. Zakharyashev. Temporal Description Logic for Ontology-Based Data Access. In *Proceedings of IJCAI* (Beijing, 3-9 August), 2013. full version is available at arXiv:1304.5185
- 2 A. Artale, R. Kontchakov, V. Ryzhikov and M. Zakharyashev. A Cookbook for Temporal Conceptual Data Modelling with Description Logics, 2012. arXiv:1209.5571.
- 3 A. Artale, R. Kontchakov, V. Ryzhikov and M. Zakharyashev. Tailoring Temporal Description Logics for Reasoning over Temporal Conceptual Models. In *Proc. of the 8th Inter-*

national Symposium on Frontiers of Combining Systems (FroCoS 2011). LNCS, vol. 6989, pages 1-11. Springer, 2011.

- 4 A. Artale, D. Calvanese, R. Kontchakov and M. Zakharyashev. The DL-Lite Family and Relations. *J. Artif. Intell. Res. (JAIR)*, 36:1-69, 2009.

3.21 ProB: Solving Constraints on Large Data and Higher-Order Formal Models

Michael Leuschel (*Heinrich-Heine-Universität Düsseldorf, DE*)

License © Creative Commons BY 3.0 Unported license
© Michael Leuschel

The B-method is a formal method for specifying safety critical systems, reasoning about those systems and generate code that is correct by construction. B is based on predicate logic, augmented with arithmetic (over integers), (typed) set theory, as well as operators for relations, functions and sequences. As such, B provides a very expressive foundation which is familiar to many mathematicians and computer scientists. One of our goals is to be able to use B as a high-level and easy to use language to express constraints in wide variety of application areas. In this talk we have presented the ProB [1] constraint solver and model checker for the B method. In particular, we describe how a variety of other formalisms and industrial problems can be mapped to B and how ProB can be used for automated reasoning. For example, we have discussed the iUML tool and its integration into the Rodin toolset and link with ProB for animation and validation. Finally, we also touch upon the various backends [2] of ProB that can be used in practice.

References

- 1 D. Plagge and M. Leuschel. Seven at a stroke: LTL model checking for high-level specifications in B, Z, CSP, and more. In *International journal on software tools for technology transfer*. vol. 12, pages 9-21. Springer, 2010.
- 2 D. Plagge and M. Leuschel. Validating B, Z and TLA+ using ProB and Kodkod. In D. Giannakopoulou and D. Méry, editors, *Proceedings FM'2012*, LNCS 7436, pages 372-386. Springer, 2012.

3.22 A Declarative Approach to Distributed Computing

Jorge Lobo (*Universitat Pompeu Fabra – Barcelona, ES*)

License © Creative Commons BY 3.0 Unported license
© Jorge Lobo

Joint work of Ma, Jiefei; Le, Franck; Wood, David; Russo, Alessandra; Lobo, Jorge
Main reference A Declarative Approach to Distributed Computing: Specification, Execution and Analysis. To appear in *Theory and Practice of Logic Programming*.

There is an increasing interest in using logic programming to specify and implement distributed algorithms, including a variety of network applications. These are applications where data and computation are distributed among several devices and where, in principle, all the devices can exchange data and share the computational results of the group. In this paper we propose a declarative approach to distributed computing whereby distributed algorithms and communication models can be (i) specified as action theories of uents and actions;

(ii) executed as collections of distributed state machines, where devices are abstracted as (input/output) automata that can exchange messages; and (iii) analysed using existing results on connecting causal theories and Answer Set Programming. Results on the application of our approach to different classes of network protocols are also presented.

3.23 Reasoning on conceptual schemas of spatial data

Stephan Maes (TU Dresden, DE)

License © Creative Commons BY 3.0 Unported license
© Stephan Maes

Main reference S. Maes, “Reasoning on class relations: An overview,” in: *Cognitive and Linguistic Aspects of Geographic Space, Lecture Notes in Geoinformation and Cartography*, pp. 237–257, Springer, 2013.

URL http://dx.doi.org/10.1007/978-3-642-34359-9_13

Spatial databases require specific integrity constraints to specify restrictions on geometric and topological relations and properties. This work focuses on integrity constraints that define cardinality restrictions for a certain instance relation (for example a topological relation) between all entities of the involved classes. These constraints also allow for automated reasoning to find contradictions and redundancies and to evaluate the compliance with application requirements. Therefore the logical properties of the applied instance relations and those of the cardinality restrictions have to be considered, in particular symmetry and compositions, but also other inferences. The presentation provides an overview of the different types of integrity constraints of spatial data, summarizes research on corresponding reasoning approaches and outlines possible future work. Most of the discussed reasoning algorithms have been implemented in a research prototype that is available at <http://www.stephanmaes.de/classrelations.html>.

References

- 1 S. Mäs, . Reasoning on class relations: an overview. In *Raubal, M., Mark, D. M., and Frank, A. U., editors, Cognitive and Linguistic Aspects of Geographic Space – New Perspectives on Geographic Information Research, Lecture Notes in Geoinformation and Cartography*, pages 237-257. Springer, 2013.
- 2 S. Mäs, and W. Reinhardt. Categories of geospatial and temporal integrity constraints. In *International Conference on Advanced Geographic Information Systems and Web Services, GEOWS 2009*, pages 146-151. IEEE Computer Society, 2009.
- 3 S. Mäs. Reasoning on spatial relations between entity classes. In *Cova, T. J., Miller, H. J., Beard, K., Frank, A. U., and Goodchild, M. F., editors, Geographic Information Science, 5th International Conference, GIScience 2008, Proceedings*, volume 5266 of *Lecture Notes in Computer Science*, pages 234-248. Springer, 2008.

3.24 On BDD, Finite controllability and the BDD/FC conjecture

Jerzy Marcinkowski (University of Wrocław, PL)

License © Creative Commons BY 3.0 Unported license
© Jerzy Marcinkowski

Main reference T. Gogacz, J. Marcinkowski, J., “On the BDD/FC conjecture,” in Proc. of the 32nd Symposium on Principles of Database Systems (PODS '13), pp. 127–138, ACM, 2013.

URL <http://dx.doi.org/10.1145/2463664.2463668>

FC (Finite controllability) and BDD (the Bounded Derivation Depth property) are two properties of Datalog/TGD programs (or of TBoxes, if this is how you wish to call them).

BDD is equivalent to Positive First Order rewritability – the very useful property that allows us to use (all the optimizations of) DBMS in order to compute the certain answers to queries in the presence of a theory.

Finite Controllability of a theory/TBox T means that if the certain answer to a query Q , for a database instance/ ABox D , in the presence of T is 'no' then this 'no' is never a result of an unnatural assumption that the counterexample can be infinite.

We conjecture that for any theory T the property BDD implies FC. We prove this conjecture for the case of binary signatures (which in particular means that it holds true for the Description Logics scenario).

3.25 On the Relationship Between OBDA and Relational Mapping

Marco Montali (Free University of Bozen-Bolzano, IT)

License © Creative Commons BY 3.0 Unported license
© Marco Montali

A position talk highlighting connections, similarities and differences between the framework of Ontology-Based Data Access, and the one of Object-Relational Mapping. Despite some key differences, such as the fact that OBDA works with incomplete information and under the assumption that there is an impedance mismatch between the conceptual model and the underlying database, while Object-Relational Mapping works with complete information and assuming a lossless connection between these two layers, we advocate the need for cross-fertilization between the two settings.

3.26 Reasoning about the Effect of Structural Events in UML Conceptual Schemas

Xavier Oriol (UPC – Barcelona, ES)

License © Creative Commons BY 3.0 Unported license
© Xavier Oriol

Joint work of Oriol, Xavier; Teniente, Ernest; Tort, Albert

In this talk, we propose an approach which is aimed at providing feedback regarding the dynamic behaviour of the schema when some set of structural events (i.e. insertion or deletion of instances) happens simultaneously in a consistent information base state.

In this way, given a conceptual schema and an IB state for that schema, we answer how can we insert or delete concrete instances of classes/associations without violating

any integrity constraint. Concretely, we answer the minimal sets of structural events that, combined with the desired insertions/deletions, lead the IB state to a new consistent one.

3.27 Information and dependency preserving BCNF decomposition algorithm via attribute splitting

Elena V. Ravve (ORT Braude College – Karmiel, IL)

License © Creative Commons BY 3.0 Unported license
© Elena V. Ravve

Joint work of Ravve, Elena V.; Makowsky Johann A.

Main reference J. Makowsky, E. Ravve, “BCNF via attribute splitting,” in: *Conceptual Modelling and Its Theoretical Foundations, LNCS, Vol. 7260*, pp. 73–84, Springer, 2012.

URL http://dx.doi.org/10.1007/978-3-642-28279-9_7

Databases are designed in an interactive way between the database designer and his client. Application driven design uses the language of Entity-Relationship modeling. 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 null values, insertion, deletion and modification anomalies, 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 . 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, 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. We introduce an additional way of restructuring databases 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. Design theory for relational databases fell out of fashion twenty years ago and few papers were published on the topic. However, with the fashionable trend of XML-driven databases, renewed interest in normal forms emerged. Hopefully, our approach can be used to formulate design criteria for XML based databases.

3.28 Semantic-based Mappings

Guillem Rull (UPC – Barcelona, ES)

License © Creative Commons BY 3.0 Unported license
© Guillem Rull

Joint work of Mecca, Giansalvatore; Rull, Guillem; Santoro, Donatello; Teniente, Ernest

Main reference G. Mecca, G. Rull, D. Santoro, E. Teniente, “Semantic-Based Mappings,” in Proc. of the 32nd Int’l Conf. on Conceptual Modeling (ER’13), pp. 11–13, Hong Kong, November, 2013.

URL <http://www.essi.upc.edu/~grull/papers/MRST13.pdf>

In classical data exchange, where data is to be transferred from a source into a target, schema mappings are specified at the database level. However, in many cases, there is a conceptual schema available for the target database, so we propose to map the source into the target conceptual schema instead of the target database. In this way, the abstraction on the details of how the data is stored in the database provided by the conceptual schema allows for a simpler mapping, which, in turn, makes easier the mapping design task. The challenge is that mappings that target a conceptual schema are not directly executable under the current techniques, so we propose an automatic rewriting algorithm that transforms the given database-to-conceptual schema mapping into a classical database-to-database one.

3.29 The Curse of Restructuring in Dependency Theory

Klaus-Dieter Schewe (Software Competence Center – Hagenberg, AT)

License © Creative Commons BY 3.0 Unported license
© Klaus-Dieter Schewe

Joint work of Schewe, Klaus-Dieter; Attila, Sali


Main reference A. Sali, K.-D. Schewe, “Weak Functional Dependencies on Trees with Restructuring,” *Acta Cybern.* 20(2), pp. 285–329, 2011.

URL http://www.inf.u-szeged.hu/actacybernetica/edb/vol20n2/Sali_2011_ActaCybernetica.xml

The talk addresses the problem that restructuring in complex value databases (used as a broad term to capture any post-relational database structures) is unavoidable, in particular, if disjoint union is permitted as a constructor. However, the consequences for dependency theory are dramatic. For instance, for (weak) functional dependencies on database structures built from record, lists, sets and multisets a nice axiomatisation can be found, whereas the addition of the union constructor leads to a vast amount of additional structural axioms for wFDs and even non-axiomatisability for FDs. However, the core of the axiomatisation remains the same in the sense that the additional axioms merely cover the properties of coincidence ideals, i.e. the set of subattributes, on which two complex values coincide. This raises the open question, whether it is possible to parameterise dependency theory by isolating such structural properties.

3.30 AuRUS: Automated Reasoning on UML Schemas

Ernest Teniente (UPC – Barcelona, ES)

License  Creative Commons BY 3.0 Unported license
© Ernest Teniente

Joint work of Rull, Guillem; Farré, Carles; Queralt, Anna; Urpí, Toni


Main reference G. Rull, C. Farré, A. Queralt, E. Teniente, T. Urpí, “Aurus: explaining the validation of UML/OCL conceptual schemas,” *Software and Systems Modeling*, 28pp., 2013.

URL <http://dx.doi.org/10.1007/s10270-013-0350-8>

The validation and the verification of conceptual schemas have attracted a lot of interest during the last years, and several tools have been developed to automate this process as much as possible. This is achieved, in general, by assessing whether the schema satisfies different kinds of desirable properties which ensure that the schema is correct. In this talk we describe AuRUS, a tool we have developed to analyze UML/OCL conceptual schemas and to explain their (in)correctness. When a property is satisfied, AuRUS provides a sample instantiation of the schema showing a particular situation where the property holds. When it is not, AuRUS provides an explanation for such unsatisfiability, i.e., a set of integrity constraints which is in contradiction with the property.

3.31 Visual reasoning with (functional) dependencies

Bernhard Thalheim (Universität Kiel, DE)

License  Creative Commons BY 3.0 Unported license
© Bernhard Thalheim

Joint work of Bernhard Thalheim; Ove Sörensen

Main reference O. Sörensen, B. Thalheim, “Semantics and pragmatics of integrity constraints,” in: *Semantics in Data and Knowledge Bases, LNCS*, Vol. 7693, pp. 1–17, Springer, 2013.

URL http://dx.doi.org/10.1007/978-3-642-36008-4_1

Logical reasoning is main basis for axiomatisation of constraints. Often first-order predicate calculus is used. Human reasoning is however often spatial. Any child knows the meaning of notions such as “behind”, “far”. Graphical presentation is used for reasoning and explanation before we learn to write. Most classes of database constraints are however expressed by logical formulas. Typical simple calculi have been developed for functional dependencies or inclusion constraints. The axiomatisation for multivalued dependencies is more difficult to understand and to use. Already the class of cardinality constraints demonstrates that numerical calculi support reasoning far simpler.

We show however that graphical reasoning is far simpler for most classes of database constraints. For instance, functional dependencies can be represented by directed graphs. These graphs allow to reason on sets of functional dependencies. The classical Armstrong axiomatisation can be extended to such graphs. These graphs allow to reason on functional and on negated functional dependencies.

These graphs can also be used for constraint acquisition. It is possible to reason on functional, negated functional and underivable functional dependencies in a simple and powerful fashion. Since normalisation assumes that all valid functional dependencies are known and this problem is exponential of the set of attributes we need more sophisticated approaches.

3.32 Validation of Complex Domain-Specific Modeling Languages

Dániel Varró (Budapest Univ. of Technology & Economics, HU)

License © Creative Commons BY 3.0 Unported license
© Dániel Varró

Despite the wide range of existing generative tool support, constructing a design environment for a complex domain-specific language (DSL) is still a tedious task as the large number of derived features and well-formedness constraints complementing the domain metamodel necessitate special handling. Incremental model queries as provided by the EMF-IncQuery framework can (i) uniformly specify derived features and well-formedness constraints and (ii) automatically refresh their result set upon model changes. However, for complex domains, such as avionics or automotive, derived features and constraints can be formalized incorrectly resulting in incomplete, ambiguous or inconsistent DSL specifications.

To detect such issues, we propose an automated mapping of EMF metamodels enriched with derived features and well-formedness constraints captured as graph queries in EMF-IncQuery into an effectively propositional fragment of first-order logic which can be efficiently analyzed by the Z3 SMT-solver. Moreover, overapproximations are proposed for complex query features (like transitive closure and recursive calls). Our approach will be illustrated on analyzing DSL being developed for the avionics domain.

We further aim to address to reason about the evolution of models (along potentially infinite state spaces by a combined validation technique based upon shape analysis). Concrete graph based models are abstracted into different kind of shapes to represent the context of model elements in an abstract way. Formal validation is supported on the level of shapes is by a combined use of (a) SMT-solvers to derive potentially relevant context of shape elements, (b) incremental instance-level model queries to filter irrelevant context and (c) some dedicated abstract libraries.

3.33 Reasoning about Dependencies in Schema Mappings

Qing Wang (Australian National University – Canberra, AU)

License © Creative Commons BY 3.0 Unported license
© Qing Wang

Schema mappings have been extensively studied in the past decades from a variety of aspects, including high-level specification languages, computational properties, optimization, etc. To discover logical consequences among source constraints, target constraints and mapping constraints of a schema mapping, we develop a graph-based framework that captures the inter-relationship between attributes of different relations. User feedback can be incorporated into the framework to minimize ambiguity in designing schema mappings. In doing so, we can characterize the class of sources instances that have a corresponding target instance under a given schema mapping, and conversely can also capture properties that all target instances for a given source instance must have.

4 Working Groups

4.1 On the Practical Applicability of Current Techniques for Reasoning on the Structural Schema

Ernest Teniente

License  Creative Commons BY 3.0 Unported license
© Ernest Teniente

There has been plenty of promising results for providing automated reasoning on the structural part of the conceptual schema and several prototype tools have been developed with this purpose. However, most of these results have remained at the academical level and the industry is not aware of them or it does not consider them relevant enough since it is not using them in software development. With the aim of reducing the gap between academy and industry, the discussion of the participants in this group was aimed at providing an answer to the following questions:

1. What do we need to convince the industry that this technology is useful?
2. Can we come up with a common vocabulary for the various research disciplines that work on this topic?
3. Can we come up with a research agenda of the problems we have to solve?

The first part of the discussion was devoted to identify the most relevant topics that should be addressed to be able to provide an answer to those questions. In particular, there was an agreement on five different topics: *identifying the relevant properties, coming up with a common agreement on the formalization (i.e. definition) of the properties, the need for explanations, the need for benchmarks, and showing the scalability of the tools developed so far*. The second part of the discussion went into digging down for each topic and trying to identify the most relevant issues that should require a proper answer to make the results in this area applicable in practice.

The outcome of the discussions for each topic is summarized in the following:

Properties

Two different kinds of properties were identified: those related to reasoning only at the schema level and properties involving both the schema and the data. The first kind of properties are aimed at detecting whether the schema being defined is correct. So, whenever one such property is not satisfied by the schema, or its results do not correspond to the ones expected by the designer, it means that the schema is not properly defined and it must be necessarily changed. The second kind of properties, i.e. those involving data, should be understood more as *services* provided by the system at run-time rather than properties denoting that the schema is ill-specified at design-time. In general, all the properties arising from the discussions are well-known problems in the area. The most relevant properties for each group are the following:

1. Properties related only to the schema
 - Schema satisfiability. There was an agreement that the empty state should not be accepted as a solution. It was also clear the need to distinguish between finite and infinite satisfiability.
 - Class and association satisfiability

- Constraints and class redundancy, in the sense that they are entailed by the rest of the schema.
 - User-defined property verification, aimed at allowing the designer to determine whether the state satisfies the requirements of the domain. One possible way to achieve it is by showing the satisfiability of a partially specified state envisaged by the designer.
2. Services involving data
- *Model checking*, i.e. whether a set of instances satisfies a set of constraints (aka integrity checking). This should be combined with techniques to “restore” consistency when the constraints are violated (or to handle with the “inconsistent” data). It is also worth noting that the database view of data should be taken for this purpose, i.e. closed world assumption: the data is a model of the constraints
 - Satisfiability checking over the data. A special effort should be devoted to deal with incomplete databases (e.g. nulls or disjunctive values). Model generation (aka integrity maintenance) is a must. In this case, the open world assumption is needed in the sense that you can invent new things.
 - Query processing
 - View updating
 - Materialized view maintenance
 - Impact analysis of an update
 - Test-data generation

Definition of the properties

There was an agreement that one of the difficulties with convincing the industry about the usefulness of these properties relies on the lack of agreement in the literature about the precise definition of most of such properties. Therefore, one of the first things the community should do is to agree with their formal definition. There was not enough time for doing this during the break out sessions but the working group identified some issues to be taken into account:

- There is a need to clarify whether finite or infinite interpretations are considered
- There is a need to clarify whether the definition uses the database view or the “open-world” view
- There is also a need to clarify which is the semantics used

Explanations

Having tools to show that all of this works was considered to be the most important general concern for showing that the previous properties can be useful in practice. In addition to being able to check these properties, these tools should also explain the results of performing automated reasoning on the conceptual schema. Specifically, it would be interesting to know what kind of explanations would the industry like to have, what is an explanation and in which language should they be shown to the designer. Moreover, explanations should abstract away from whatever logic is used underneath and they should be given regarding to the model the user is referred to. Facilities for what-if scenarios could also be interesting for the industry. The working group identified also some possible kinds of explanations, making a distinction when the property under validation is satisfied or it is not.

- The property is satisfied
 - Instance or snapshot or witness (generated example)

- An abstraction of the proof (wrt the terminology of the model)
- The property is not satisfied
 - Providing the (minimal) set(s) of constraints that give raise to the violation
 - An abstraction of the proof (wrt the terminology of the model)
 - Causal reasoning, i.e. suggestions about how to repair the violation

Benchmarks

Benchmarks are very important for industry. However, little attention has been paid to them in the area. In fact, there is not yet an agreement on what a benchmark for automated reasoning on conceptual schemas should be. The working group considered that such benchmarks should at least include a motivation underlying the benchmark (i.e., why is this benchmark for); the schema (and the data, if necessary) under consideration; the properties to be checked by the benchmark and their expected output. The definition language of the benchmark (EER, UML/OCL, ORM, etc.) and the semantics used in the benchmark should also be clearly stated. Additionally, some questions and ideas arised as a result of the discussions:

- How many benchmarks do we need?
 - It depends on the purpose of the benchmark. Scalability vs language expressivity, for instance.
 - Could we come up with a repository of benchmarks?
- Benchmarks for education seem interesting
- Establishing fair benchmarks
 - Separation of concerns and avoiding conflict of interests are important issues
 - Having a contest for this community could be interesting
- Evaluation criteria are also needed

Scalability

There was a clear agreement that scalability has to be necessarily addressed to convince the industry. Moreover, there seemed to be a common understanding that it would probably be already covered if properties and benchmarks were correctly defined. Other aspects arising from the short discussion we had on this topic were:

- Large data sets vs large complex schemata
- Scalability is not only performance
- Visualizing large-schemas properly

Conclusions

The working group agreed that there is still a lot of things to do for convincing the industry about the practical applicability of current techniques for reasoning on the structural schema. Most of these things have been summarized along this section. However, the promising results achieved so far and the existence of several prototype tools that can be applied in practice allow us to be optimistic about the achievement of this ambitious goal. Having practical tools to show that all of this works was agreed to be a necessary condition for this purpose.

Participants:

- Achim D. Brucker (SAP Research – Karlsruhe, DE)
- Alessandro Artale (Free University of Bozen-Bolzano, IT)
- Alessandro Mosca (Free University of Bozen-Bolzano, IT)
- Bernhard Thalheim (Universität Kiel, DE)
- Carolina Dania (IMDEA Software Institute, ES)
- David W. Embley (Brigham Young University, US)
- Ernest Teniente (UPC – Barcelona, ES)
- Ingo Feinerer (TU Wien, AT)
- Mirco Kuhlmann (Universität Bremen, DE)
- Parke Godfrey (York University – Toronto, CA)
- Sophie Dupuy-Chessa (LIG – Grenoble, FR)
- Xavier Blanc (University of Bordeaux, FR)

4.2 Reasoning about the Conceptual Schema Components Capturing Dynamic Aspects

Diego Calvanese

License © Creative Commons BY 3.0 Unported license
© Diego Calvanese

The discussion in the working group started from the observation that the topic to be addressed is quite challenging for a variety of reasons. Indeed, there is a consensus about the key aspects that are of importance and need to be considered when modeling the structural aspects of a system and when reasoning over such a conceptualization. Instead, there was a consensus that when it comes to modeling and reasoning over the dynamic aspects of an information system, and hence its evolution over time, the situation is much less clear and not at all consolidated, both with respect to the properties to be modeled, and with respect to the formalisms to be adopted.

After a round in which each participant briefly introduced what it considered important aspects to be tackled in the discussion, the working group set up an ambitious agenda comprising the following list of points and questions that it intended to address, ordered by importance:

1. Which dynamic and/or temporal properties should be modeled? How should the structural and dynamic components be combined?
2. Which modeling formalisms, possibly based on logic, should be adopted for capturing the dynamic together with the static aspects of a system? In addition to the expressive power of the formalism, also the aspects related to the computational complexity and hence efficiency of reasoning should be taken into account, and hence discussed. Possibly, tractable fragments should be identified.
3. Identify specific problems, use cases, and scenarios related to dynamic aspects that come from industrial requirements (e.g., security).
4. Identify important design-time tasks where reasoning about dynamic aspects is of importance (e.g., exploratory design, analysis, planning, synthesis, verification).
5. Identify important run-time tasks where reasoning about dynamic aspects is of importance (e.g., analysis, validation, monitoring, mining).
6. Discuss the different levels of abstraction of models and their implementation.

When the group set out to discuss Item 1 of the above list, it became immediately clear that there was a very tight connection between the dynamic/temporal properties and the formalism to adopt for modeling them, so that Items 1 and 2 were actually discussed together. In fact, getting a clarification on these two points was considered almost a prerequisite for

addressing further issues, so that the discussion on them took up almost the whole time available to the group. In the end, Items 3 and 6, although considered important, could not be addressed. Items 4 and 5 dealing respectively with the design-time and run-time tasks that could be supported by reasoning over dynamic aspects were considered and discussed together.

We summarize below the outcome of the discussions that took place in the group.

Properties and types of formalisms

There was an agreement that a convenient and general way to characterize the semantics of systems that evolve and change over time is by means of labeled transition systems over first-order structures (i.e., databases). Several dimensions for characterizing the formalisms that are able to specify and capture dynamic aspects were identified, specifically:

- adoption of a linear vs. a branching time model and formalism to specify dynamic properties;
- adoption of a propositional abstraction vs. first-order formalisms that are able to quantify over a single state of the (transition) system vs. first-order formalisms that are able to quantify across different states;
- how data should be incorporated: adoption of a pure first-order model vs. first-order model with built-in data types vs. first-order model enriched with additional structure, e.g., arithmetics;
- adoption of a discrete vs. a real-time model;
- adoption of a model based on a sequence/tree of states (i.e., databases) that are queried independently vs. a temporal database like model, where one can query arbitrarily the sequence/tree of states, also doing kind of complex joins across states;
- consideration or not of deontic and organizational aspects

Usability was considered an important dimensions that will have a strong impact on the choices of the formalism, but is an aspect that is orthogonal to all the ones mentioned above.

Concrete formalisms

The working group discussed also several concreted formalism that could be adopted for modeling and reasoning over dynamic aspects. Specifically, the following was considered and discussed:

- First-order mu-calculus was identified as a very general and powerful formalism for specifying dynamic properties.
- First-order variants of temporal logics, such as LTL, CTL, PDL, TLA+, B, and LDL were also mentioned; the availability of corresponding model checking tools (e.g., TLC) was considered an important aspect to take into account.
- Decidable fragments of the above languages should possibly be adopted. An example are temporal description logics, where to obtain decidability of reasoning over the combination of static and dynamic aspects (in some cases severe) restrictions on the expressive power of the modeling formalisms are made.
- Action-based languages that allow one to model processes, e.g., by means of rules were mentioned.
- It was observed that OCL, which combined with UML is already widely used to capture structural constraints of a system, can also be adopted to specify pre- and post-conditions of actions.

Interaction between the static and dynamic components

An important aspect was considered to be how the models and formalisms for the static and for the dynamic parts of the system interact with each other.

- One possibility is that the static and the dynamic aspects are part of the same conceptual model, which becomes a so-called temporal conceptual model.
- A different approach is to model the two aspects separately and then deal with their connection, which deals results in a structural conceptual model plus a business process.
- Also the form of the dynamic queries that can be posed over the evolution of the static component over time needs to be taken into account.

Conclusions

The working group agreed that there is a lack of a reference framework encompassing all relevant aspects. While such a reference framework on the one hand might be too general in practice, on the other hand it could at least provide a way to connect the different approaches. It was also observed that when we consider models for the dynamics, there is even a lack of understanding and agreement of when a model/formalism should be called “conceptual”, and what the right level of abstraction for a “dynamic conceptual model” actually is. Similarly to the case of modeling and reasoning over the structural aspects of a system, also in this case a key aspect is the trade-off between generality and expressive power on the one hand and decidability and complexity of inference on the other hand.

Participants:

- | | | |
|---|--|--|
| ■ C. Marijke Keet (University of KwaZulu-Natal – Durban, ZA) | ■ Geri Georg (Colorado State University, US) | ■ city Bozen-Bolzano, IT) |
| ■ Carsten Lutz (Universität Bremen, DE) | ■ Jerzy Marcinkowski (University of Wrocław, PL) | ■ Martin Gogolla (Universität Bremen, DE) |
| ■ Dániel Varró (Budapest Univ. of Technology & Economics, HU) | ■ Jorge Lobo (IBM TJ Watson Research Center – Hawthorne, US) | ■ Michael Leuschel (Heinrich-Heine-Universität Düsseldorf, DE) |
| ■ Diego Calvanese (Free University Bozen-Bolzano, IT) | ■ Marco A. Casanova (PUC – Rio de Janeiro, BR) | ■ Michael Zakharyashev (Birkbeck College London, GB) |
| | ■ Marco Montali (Free Univer- | ■ Xavier Oriol (UPC – Barcelona, ES) |

4.3 New Challenges for Automated Reasoning on Conceptual Schemas

Sven Hartmann

License © Creative Commons BY 3.0 Unported license
© Sven Hartmann

Our working group started with a brainstorming session where everybody contributed ideas and opinions about emerging trends and challenges in our field of research. Soon the discussion focussed around the following set of questions:

- Which major trends are underway in database modelling and management? What will be the trends of tomorrow? Which trends will have a lasting impact?
- What challenges arise for the conceptual modelling of databases? What contributions can the conceptual modelling community make? What new research directions emerge for the conceptual modelling community?

- What support can automated reasoning provide to tackle these challenges? What concepts, theories, and methods must be developed by the automated reasoning community to enable solutions? What new research directions emerge for the automated reasoning community?

After a lively exchange of thoughts a range of trends and issues for research were identified as important and suggested for further discussion. These can be grouped into the following topic areas:

1. Big data and the exploration of conceptual schemas
2. Integration of database and software engineering formalisms and methods
3. Co-evolution of schemas and views
4. Updatable views
5. Assumptions of reasoning tasks
6. Meta modelling
7. Research integration
8. Benchmarks
9. Quality assurances and cost models

During the discussion in our group it soon became apparent that some of the upcoming research issues are extensions of established problems that have motivated research on automated reasoning for conceptual schemas in the past and still await answers that unlock them for uptake by database practitioners.

Big Data and Schema Exploration

Big Data is one of today's mega trends that leads to a multitude of new research issues. The group shared opinions on whether and how conceptual modelling can help to understand and process massive datasets of heterogeneous data. The following items were suggested for further investigation:

- Can we use automated reasoning techniques to extract / adjust / enhance conceptual information from big data (such as sensor data or stream data)?
- How would that be different from data mining?
- How can existing conceptual models be combined with data mining techniques to improve concept models?
- What formalisms are needed to capture conceptual information in big data.

Database vs software development

In view of the increasing complexity of future information systems the group found it necessary to develop and deploy research-led strategies for integrating formalisms, methods and practices used in database and software development.

- How can constraint checking efforts in DBMS and application software be integrated?
- What guarantees do software developers need about databases?
- Does correctness / completeness really matter?
- How can these guarantees be best communicated?
- Do we need to map them to implementation code?
- How to establish a global perspective on constraint enforcement?

Co-evolution of Schemas and Views

The group discussed challenges that arise from changing information needs in complex information systems. There was also agreement that conceptual modelling in practice has diverse stakeholders that require tailored views on databases during the entire life cycle.

- How can conceptual changes of views be propagated to the underlying schema?
- What updates can be supported?
- How can schema/view mappings best evolved?
- What constraints can be supported?
- What criteria need to be considered?
- How can automated reasoning help?
- Can we understand and contribute to tools like Hibernate?

Updatable views

The discussion addressed research challenges for automated reasoning that arise when databases are updated by diverse users and application programs through tailored views.

- How can data updates on views be propagated to the underlying conceptual schema?
- What updates can be supported?
- What constraints can be supported?
- What criteria need to be considered? (lossless, inference-proof, ...)
- How can automated reasoning help?

Assumptions of reasonings tasks

The group discussed the discrepancy between the challenges that database designers face in practice and the answers that the research community is able to provide, and asked for reasons.

- How to bridge the mismatch between the relational model and SQL?
- Can our methods handle partial information?
- Can our methods handle duplicates?
- How can SQL features be handled in the logic languages that we use?

Meta modelling

In view of the diversity of conceptual modelling frameworks in use the group agreed that automated reasoning can only be successful if the underlying semantics is revealed and formalized. The opportunities of model-based approaches and schema translations were discussed, too.

- Do we need semantics for meta models?
- Can we find and justify patterns for model transformations in conceptual modelling?
- Are purely syntactical transformations desirable / achievable?
- How can graphical and textual languages for schema declaration be used simultaneously to provide optimal support for designers?

Research integration

There was agreement that the tackling emerging research challenges would benefit from joint efforts of the community which requires a common understanding and extended collaboration.

- Do we understand each other? Even if we know about different terminologies, are we always aware of the one we are using?
- We use different methods for constraint handling, different languages for declaring constraints, different constraint classes – how can these efforts be integrated?
- Do we need new approaches to tackle the conceptual complexity of emerging applications?
- Can modelling and reasoning be combined with simulation?

Benchmarks

We discussed what ingredients would be most helpful to facilitate future collaboration in research, and identified commonly accepted benchmarks as one important enabling tool.

- Are there proper benchmarks for conceptual modelling tasks?
- What qualifies them as benchmarks?
- How to make test cases transparent and applicable by the community?

Quality assurances and cost models

Members of our group asked what obstacles hamper the uptake of new research achievements into practice, and what is needed to overcome them. Quality assurances in the vein of service-level agreements were considered as one promising approach and suggested for investigation.

- Is it necessary / possible / desirable to predict the impacts of design decisions at conceptual level on the implementation level?
- What kind of cost models do we need / want?
- What assurances can be provided to database users?
- Is it enough to say “we do it best possible” or do we need quantitative statements?

Participants:

- | | | |
|---|--|--|
| ■ Elena V. Ravve (ORT Braude College – Karmiel, IL) | ■ Klaus-Dieter Schewe (Software Competence Center – Hagenberg, AT) | ■ Stephan Mäs (TU Dresden, DE) |
| ■ Enrico Franconi (Free University Bozen-Bolzano, IT) | ■ Mira Balaban (Ben Gurion University – Beer Sheva, IL) | ■ Stephen J. Hegner (University of Umeå, SE) |
| ■ Guillem Rull (UPC – Barcelona, ES) | ■ Qing Wang (Australian National University, AU) | ■ Sven Hartmann (TU Clausthal, DE) |
| ■ Joachim Biskup (TU Dortmund, DE) | ■ Roman Kontchakov (Birkbeck College London, GB) | ■ Thomas Baar (Hochschule für Technik und Wirtschaft – Berlin, DE) |

5 Seminar program

The program of the different sessions is given below.

Session 1: Reasoning on the Structural Schema (I)

- UML class diagrams – decision, identification and repair of correctness and quality problems, *Mira Balaban*
- OCL2FOL: Using SMT solvers to automatically reason on conceptual schemata with OCL constraints, *Carolina Dania*
- Reasoning techniques for conceptual models, *Alessandro Artale*
- Toward an ontology-driven unifying metamodel for UML Class Diagrams, *Maria Keet*

Session 2: Reasoning on the Structural Schema (II)

- “Automating reasoning on conceptual schemas” in FamilySearch—a large-scale reasoning application, *David W. Embley*
- Incremental inconsistencies detection with low memory overhead, *Xavier Blanc*
- Constraints on class diagrams, *Ingo Feinerer*
- At SAP, class models are rarely used as they are “too close to real code”, *Achim D. Brucker*

Session 3: Reasoning on the Structural Schema (III)

- Reasoning in ORM, *Enrico Franconi*
- Exploring UML and OCL Model Properties with Relational Logic, *Martin Gogolla*
- AuRUS: Automated reasoning on UML schemas, *Ernest Teniente*
- ProB: Solving constraints on large data and higher-order formal models, *Michael Leuschel*

Session 4: Extensions

- Preliminary report on an algebra of lightweight ontologies, *Marco A. Casanova*
- Temporal extended conceptual models, *Roman Kontchakov*
- Reasoning on conceptual schemas of spatial data, *Stephan Mäs*
- Validation of complex domain-specific modeling languages, *Dániel Varró*

Session 5: Reasoning about the Dynamics

- View design for updates, *Stephen Hegner*
- Reasoning about the effect of structural events in UML conceptual schemas, *Xavier Oriol*
- Automated reasoning for security and compliance properties of business processes, *Achim D. Brucker*
- Unified approaches for modeling and reasoning over processes and data, *Diego Calvanese*

Session 6: New Challenges

- The curse of restructuring in dependency theory, *Klaus-Dieter Schewe*
- A declarative approach to distributed computing, *Jorge Lobo*
- On BDD, finite controllability and the BDD/FC conjecture, *Jerzy Marcinkowski*
- Metrics for visual notations, *Sophie Dupuy-Chessa*

Session 7: Reasoning about Mappings

- Reasoning about dependencies in schema mappings, *Qing Wang*
- Semantic-based mappings, *Guillem Rull*
- Relationship between approaches to ontology-based data access and object relational techniques, *Marco Montali*
- Armstrong instances as an aid for automated reasoning, *Sven Hartmann*

Session 8: Reasoning about Dependencies

- Information and dependency preserving BCNF decomposition algorithm via attribute splitting, *Elena V. Ravve*
- Visual reasoning with (functional) dependencies, *Bernhard Thalheim*
- Representation of instance-derivations based on dependencies, *Joachim Biskup*
- Reasoning over order dependencies for relational schema, *Parke Godfrey*

Participants

- Alessandro Artale
Free Univ. of Bozen-Bolzano, IT
- Thomas Baar
Hochschule für Technik und
Wirtschaft – Berlin, DE
- Mira Balaban
Ben Gurion University – Beer
Sheva, IL
- Joachim Biskup
TU Dortmund, DE
- Xavier Blanc
University of Bordeaux, FR
- Achim D. Brucker
SAP Research – Karlsruhe, DE
- Diego Calvanese
Free Univ. of Bozen-Bolzano, IT
- Marco A. Casanova
PUC – Rio de Janeiro, BR
- Carolina Dania
IMDEA Software Institute, ES
- Sophie Dupuy-Chessa
LIG – Grenoble, FR
- David W. Embley
Brigham Young University, US
- Ingo Feinerer
TU Wien, AT
- Enrico Franconi
Free Univ. of Bozen-Bolzano, IT
- Geri Georg
Colorado State University, US
- Parke Godfrey
York University – Toronto, CA
- Martin Gogolla
Universität Bremen, DE
- Sven Hartmann
TU Clausthal, DE
- Stephen J. Hegner
University of Umeå, SE
- C. Maria Keet
University of KwaZulu-Natal –
Durban, ZA
- Roman Kontchakov
Birkbeck College London, GB
- Mirco Kuhlmann
Universität Bremen, DE
- Michael Leuschel
Heinrich-Heine-Universität
Düsseldorf, DE
- Jorge Lobo
Universitat Pompeu Fabra –
Barcelona, ES
- Carsten Lutz
Universität Bremen, DE
- Stephan Mäs
TU Dresden, DE
- Jerzy Marcinkowski
University of Wroclaw, PL
- Marco Montali
Free Univ. of Bozen-Bolzano, IT
- Alessandro Mosca
Free Univ. of Bozen-Bolzano, IT
- Xavier Oriol
UPC – Barcelona, ES
- Elena V. Ravve
ORT Braude College –
Karmiel, IL
- Guillem Rull
UPC – Barcelona, ES
- Klaus-Dieter Schewe
Software Competence Center –
Hagenberg, AT
- Ernest Teniente
UPC – Barcelona, ES
- Bernhard Thalheim
Universität Kiel, DE
- Dániel Varró
Budapest Univ. of Technology &
Economics, HU
- Qing Wang
Australian National Univ., AU
- Michael Zakharyashev
Birkbeck College London, GB

