


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