Towards a Core ORM2 Language (Research Note)

Enrico Franconi and Alessandro Mosca

Free University of Bozen-Bolzano, KRDB Research Centre, Italy
{franconi,mosca}@inf.unibz.it

Abstract. The introduction of a provably correct encoding of a fragment of ORM2 (called ORM2$^{\text{zero}}$) into a decidable fragment of OWL2, opened the doors for the definition of dedicated reasoning technologies supporting the quality of the schemas design. In this paper we discuss how to extend ORM2$^{\text{zero}}$ in a maximal way by retaining at the same time the nice computational properties of ORM2$^{\text{zero}}$.

1 ORM2$^{\text{zero}+}$: The Core Fragment of ORM2

In [1,2] we introduced a fragment of ORM2, called ORM2$^{\text{zero}}$, that captures the most frequent usage patterns of the conceptual modelling community. The language has a formal semantics specified in FOL, and a provably correct (bidirectional) encoding in the description logic $\mathcal{ALCQI}$ [1]. $\mathcal{ALCQI}$ is a decidable fragment of the OWL2 Web ontology language (a complete introduction of the syntax and semantics of $\mathcal{ALCQI}$ can be found in [3]), for which optimised very efficient tableaux-based reasoning algorithms and tools have been developed. The encoding of ORM2$^{\text{zero}}$ in description logics provably preserves satisfiability and entailment, and so the reasoning in the description logic (such as the strong and weak satisfiability of the schema, of the entities and of the predicates, and the entailment of new or stricter constraints) can be transposed back to ORM2$^{\text{zero}}$. In [1,2] we extensively comment on the limits and the incorrectness of alternative similar proposals.

Figure 1 introduces the list of the constraints that are natively present in ORM2$^{\text{zero}}$, together with their graphical notations. The table also reports about the restrictions we need to impose on the applicability of these constraints in order to preserve the decidability of the language and the feasibility of the encoding, so that we can rely on available reasoning tools. In ORM2$^{\text{zero}}$ we can express typing constraints, simple mandatory constraints, internal frequency constraints restricted to single roles, arbitrary subtyping constraints, and subset and exclusion constraints restricted to pairs of whole predicates. Note that in ORM2$^{\text{zero}}$ the subtyping is not strict (namely, subtyping is interpreted as a subset-or-equal relation between entities), and that root entities (without supertypes) are not mutually disjoint.

Despite the apparent weakness of the ORM2$^{\text{zero}}$ language, suitable combinations of the ORM2$^{\text{zero}}$ constructs can be used to encode a number of other relevant ORM2 constraints, thus extending the expressive power of ORM2$^{\text{zero}}$ to what we call ORM2$^{\text{zero}+}$. The following constraints are in ORM2$^{\text{zero}+}$: unrestricted objectification; reference

Y.T. Demey and H. Panetto (Eds.): OTM 2013 Workshops, LNCS 8186, pp. 448–456, 2013.

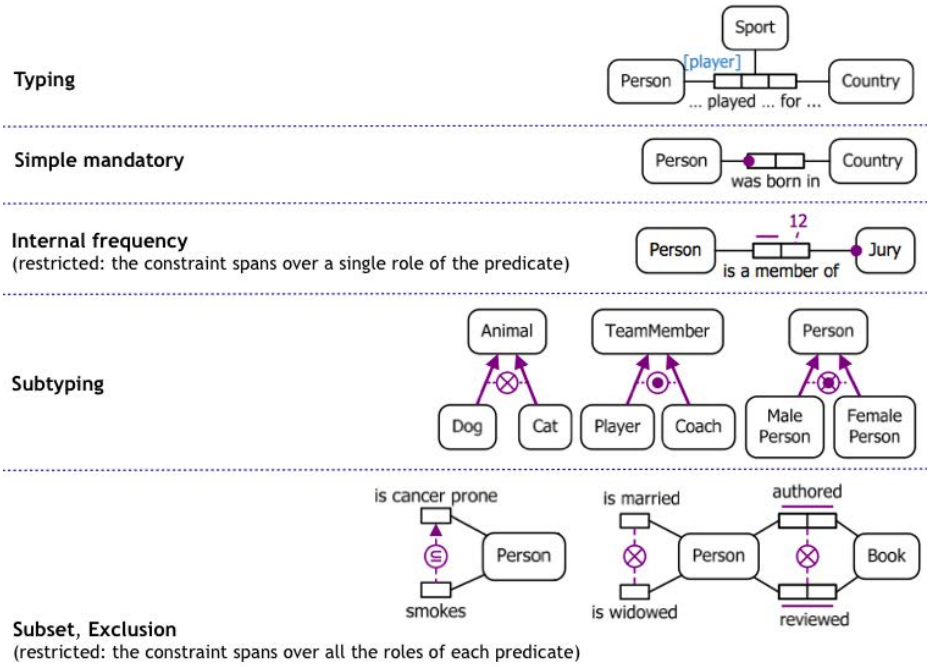


Fig. 1. List of primitive ORM2^{zero} constraints by means of examples from [4]

modes; unrestricted internal and external uniqueness and frequency constraints; unrestricted and strict subtyping; subset, exclusion, inclusive-or, exclusive-or and equality constraints restricted to pairs of single roles and to pairs of whole predicates; and the *primitive entity types disjointness* assumption. This paper describes the way how these constraints can be reduced to combinations of the primitive constraints of ORM2^{zero}. Most notably ORM2^{zero+} misses with respect to ORM2: value constraints; cardinality constraints; subset, exclusion, inclusive-or, exclusive-or and equality constraints restricted to pairs of sequences of roles of length different from one and from the arity of the involved same-arity predicates; ring constraints. If present, value, cardinality, and ring constraints would invalidate the correctness of the encoding of objectification, while unrestricted subset, exclusion, inclusive-or, exclusive-or and equality constraints would lead to undecidability of reasoning (due to the undecidability of unrestricted functional and inclusion dependencies). To complete the picture, we are exploring the possibility to add to ORM2^{zero+} derivation rules and deontic constraints.

All the ORM2^{zero+} constraints considered (with the exception of objectification) are logically equivalent to their ORM2^{zero} encodings, namely they constrain the world in the very same way and they identify the same legal databases (i.e., they have the same logical models). The encoding of an objectification constraint is weaker whenever in the schema it is necessary to have *both* the original n-ary predicate *and* the entity representing the objectified predicate *together* with its n binary predicates representing the reified roles of the original n-ary predicate. In this case the encoding is only

preserving satisfiability and entailment, but it does not preserve the models. As far as schema reasoning is concerned, this is not a limitation, since schema reasoning is based on satisfiability and entailment.

In what follows, the ORM2$^{\text{zero}}$ reductions are introduced by means of examples, the majority of which have been grasped from [4]. Many reductions are already well known in the ORM community, and some have been already presented in [4].

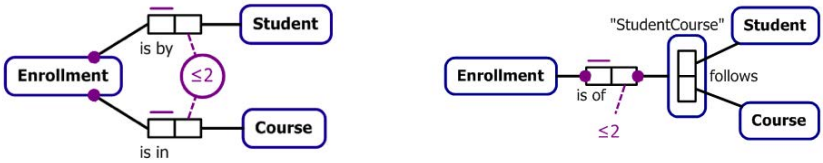
We assume that: a countable number of new fresh entity types $\mathsf{TOP}, \mathsf{TOP1}, \ldots, \mathsf{TOP}n$ are part of the language, where n is the maximum arity of predicates in the schema; TOP is a supertype of all the entity types in the schema; $\mathsf{TOP}_1 \cdots \mathsf{TOP}_i$ are supertypes of all the objectified predicates of arity i, for each $1 \leq i \leq n$; each $\mathsf{TOP}, \mathsf{TOP1}, \ldots, \mathsf{TOP}n$ is covered by its subtypes. The schemas presented in this paper may not have these assumptions explicitly written.

Reference Mode Constraint

(a) (b)

The schema (a) on the left making use of a reference mode constraint can be reduced in ORM2$^{\text{zero}}$ as shown in the schema (b) on the right. This is well known from standard ORM2.

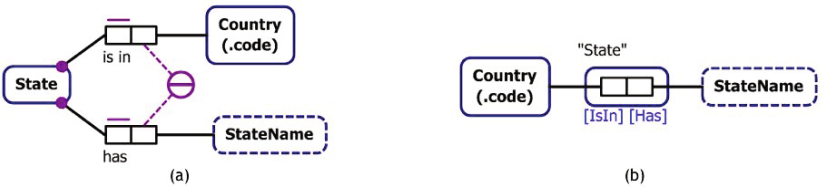
External Frequency Constraint



(a) (b)

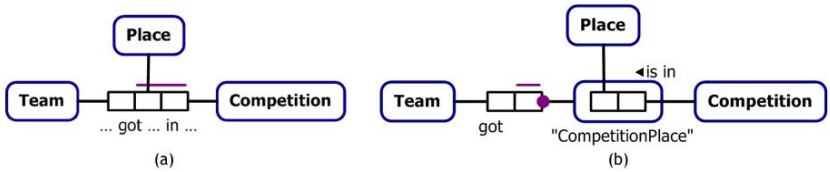
The case of the external frequency constraint is of particular interest. The external frequency is a frequency constraint that applies to roles from different predicates and its meaning relies on the implicit join operation. The main idea behind its encoding in ORM2$^{\text{zero}}$ is to make use of the objectification in order to represent the functional relationships between the tuples and the identified individuals. In the example above, Enrollment is connected with mandatory and uniqueness constraint to a new predicate whose second role is played by identifiers of (Student, Course) pairs. It is obvious that, if a model exists for the ORM2 schema, then in this model no enrolment can be repeated, and a student may enrol in the same course at most twice. The very same situation is forced by the internal frequency spanning over the second role of the new introduced predicate is of: each identifier of a (Student, Course) tuple can play that role twice (remember that the internal frequency spanning over single roles is among the primitive constructs of ORM2$^{\text{zero}}$).

External Uniqueness Constraint



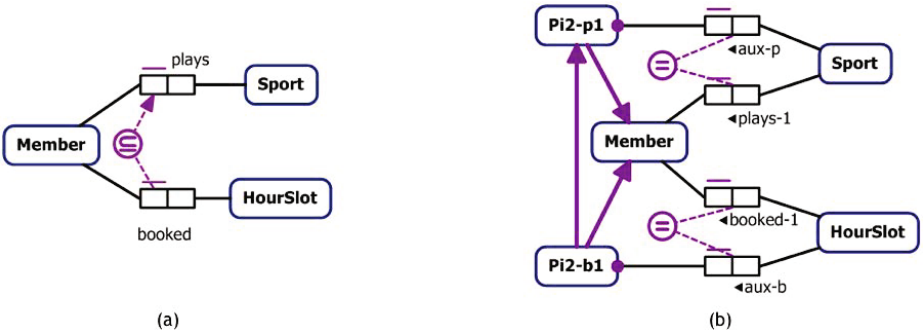
In the example, the external uniqueness constraint says that each State is univocally determined by a pair (Country, StateName). In other terms, the relationship between a pair (Country, StateName) and State is the same holding between a objectified binary tuple and its identifier: it is a bijection, indeed. Therefore, the external uniqueness in ORM2^{zero} reduces to building a new fresh predicate that, once objectified, is used to identify the individuals of State. In figure (b), State is indeed represented as the objectification of IsIn and Has. Alternatively, one can observe that uniqueness (both external and internal) is a special case of the frequency constraint "$= 1$", and therefore one could apply the reductions for frequency constraints.

Internal Uniqueness and Frequency Constraint



In ORM2, an internal uniqueness says that individuals of a given type can play a role only once. Moreover, for an 'elementary n-ary association' in ORM2, each internal uniqueness constraint must span over n-1 roles. In the figure above, the internal uniqueness forbids to record for ties: two different teams cannot occupy the same place in the same competition. Again, the reduction to ORM2^{zero} of this constraint is based on an objectification: a new predicate is introduced and objectified in order to explicitly identify the pair (Place, Competition), and its objectification is then linked with Team by means of the new predicate symbol got. The fact that a pair (Place, Competition) cannot be related twice with an instance of Team is finally represented by an internal uniqueness spanning over a single role, that is, by a native ORM2^{zero} constraint. Notice that ORM2^{zero} does not introduce internal uniqueness spanning over entire predicates because of its set-based semantics. A very similar reduction can be applied for internal frequency constraints, by replacing in the example the uniqueness constraint with the frequency constraint.

Unary Subset, Exclusion, Inclusive-or, Exclusive-or and Equality Constraints



Let's first recall that in ORM2^{zero+} unary subset, exclusion, inclusive-or, exclusive-or and equality constraints are allowed in addition to the subset, exclusion and equality constraints involving whole predicates allowed in ORM2^{zero}. The idea of this encoding for the general case is based on generating entities as the projections of the roles involved. In the example above, the entities *Pi2-p1* and *Pi2-b1* are are the projections of the two original binary predicates over the first role. The unary boolean constraints (subset, exclusion, inclusive-or, exclusive-or and equality) over those single roles are then encoded with the corresponding constraint among the two subentities representing the projections – in this example a *subset* between the two roles becomes a subtype between the two subentities. The inability of ORM2 to reuse the same role for multiple typings obliges us to introduce an additional equal copy of each involved predicate; equality among predicates is encoded in ORM2^{zero} as a cycle of *subset* between the predicates.

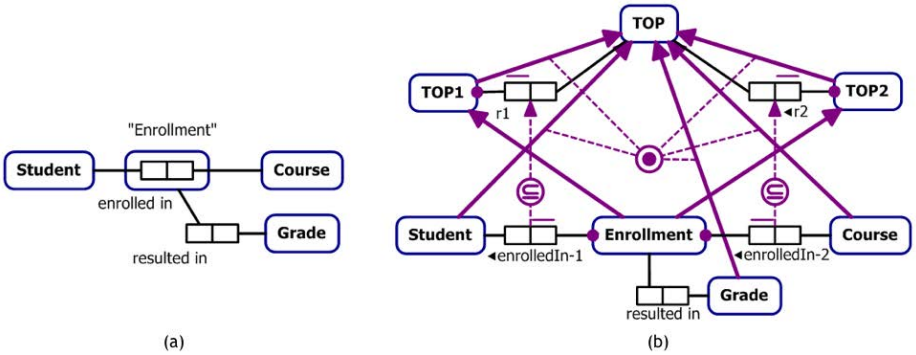
Strict Subtyping

In ORM2 there is the assumption that for each entity A being a subtype of an entity B there at least a legal database state (i.e., satisfying all the constraints in the conceptual schema) such that the extension of A in that state is different from the extension of B in that state, namely the extension of A is *stricly* a subset of the the extension of B. In other words, it is impossible to have only legal database states in which the extension of A is equal to the extension of B. In ORM2^{zero} this is achieved by checking that no cyclic subtyping among any distinct pair of entities is entailed by the schema. This check is decidable and very efficient in ORM2^{zero}.

Primitive Entity Types Disjointness

According to the ORM2 modelling methodology, in any domain (or 'Universe of Discourse') there is always a top-level partitioning of its entities into exclusive types. These entities without supertypes are called 'primitive entity types', and ORM2 assumes that *primitive entity types never overlap*. We observe that the (default) disjointness between the primitive entity types can be easily represented in ORM2^{zero} by exploiting the introduction of the TOP element as the common supertype of all the types in a ORM2^{zero} schema: only top level entities are explicitly set as subentities of TOP, and they are also said to be exclusive.

Objectification



The figure (a) above shows an ORM2 schema representing the objectification of a fact type as an entity type whose instances can play the resulted in role. The reduction of the schema in ORM2^{zero} is shown on the right side. There is a new entity "Enrolment" representing the objectified predicate, connected with two functional binary predicates objectifying the roles of the original predicate. The objectified predicate is a subtype of both TOP1 and TOP2, representing the set of the identifiers of all tuples having at least one or two roles respectively. The objectified roles in the objectified predicate are subset of the general argumental binary predicates $r1$ and $r2$. In this way, a tuple in a legal database of the schema is always represented by a unique individual (the identifier) having one functional role for each tuple component.

The ORM2^{zero} reduction above fixes the one of theorem **N/CR** (*nest/coreference*) in [4]: according to our encoding a tuple identifier is *global* to the schema, rather than *local* to the context of a predicate, thus satisfying the formal semantics given to objectification. The *local* semantics for tuple identifiers from [4] may lead to wrong conclusions. Let's consider the example schema in Fig. 2. With tuple identifiers local to an objectified predicate, a legal database exists with non-empty binary predicates "... *has Spouse* ..." and "... *is BloodRelative of* ..." one including the other and with

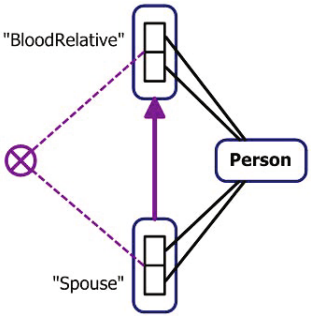


Fig. 2. The entity "Spouse" is not inconsistent with *local* semantics for tuple identifiers

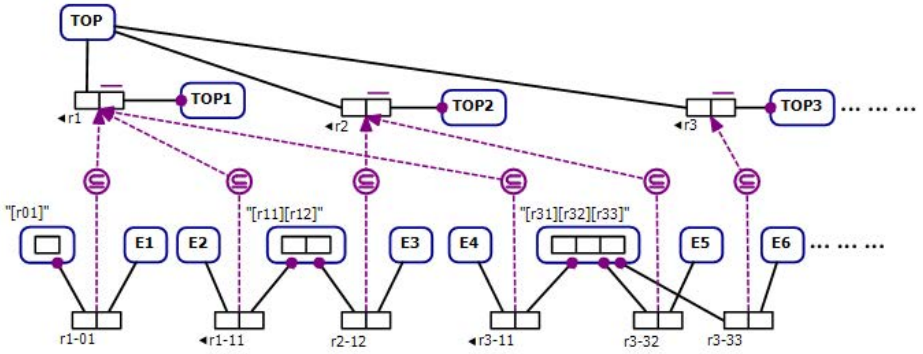


Fig. 3. Encoding of objectified predicates; the subtypes of the general TOP_i entities are not explicitly shown

non-empty disjoint "*Spouse*" and "*BloodRelative*" entities. Such a legal database should not exist, and indeed it does not exist with global tuple identifiers.

The global interpretation comes from the introduction of TOP1, TOP2, ... – each one identifying a component of a tuple by means of the objectified roles $r1, r2, \ldots$ – and by the reuse of the common predefined set of objectified roles. Thus, the identifier of a binary tuple is of type TOP1 and TOP2 (by subtyping), and it is univocally determined by a given tuple of TOP individuals. Now, the fact that the objectified roles linking TOP1 and TOP2 are the same used in the objectification of all the predicates in the schema, does the rest: a tuple (a, b), identified by t in TOP1 and TOP2, is identified in the very same way by all the objectified predicates in the schema. A general pattern for the encoding is shown in figure 3.

For those who are familiar with the 'coreference' interpretation of the objectification presented in [4], it should be evident that in the ORM2$^{\mathsf{zero}}$ reduction the external uniqueness constraints (forcing the objectified entity to be coreferenced by the pair of linked types) simply disappeared. Indeed, the external uniqueness can be safely removed in the ORM2$^{\mathsf{zero}}$ representation of the objectification because of the semantics the language. The crucial bit to understand the redundancy of this uniqueness in ORM2$^{\mathsf{zero}}$ relates to the so-called 'relation-descriptiveness' of the involved models, a model-theoretic property that have been discussed in, e.g., [5] for formal languages having the 'tree model property' (as, for example, the logic $\mathcal{ALCQI}$). Without entering into the technical details, a model is said to be relation-descriptive if each tuple in an objectified predicate is represented by one and only one individual. Formal results then show that if one is interested in reasoning on schemas represented in languages showing the relation-descriptive property, at least one of such a model is always available. Since ORM2$^{\mathsf{zero}}$, and its logical counterpart, enjoy both the relation-descriptive property, external uniqueness constraints are no longer necessary to enforce a bijection between identifiers and objectified tuples. Because of this, there is a theorem stating that this encoding for objectification does preserve satisfiability and entailment, but does not necessarily preserve the structure of the legal databases (the models). As we claimed before, this is not very relevant for our purposes, which are about reasoning over the schema.

Whenever there are additional constraints on the roles of an objectified predicate, we need a strategy for adequately encoding them to the target ORM2^{zero} schema: given an ORM2^{zero} schema with an objectified predicate, (a) copy all the unary frequency and mandatory constraints in some role of the objectified predicate to the corresponding binary predicate representing the objectified role in the target schema, (b) copy the subset and exclusion constraints in the objectified predicate to the corresponding entity, possibly continuing by objectifying also other predicates involved in these constraints.

2 Concluding Remarks

The complete picture thus shows that ORM2^{zero} is an extremely powerful conceptual modelling language. As we have already said, ORM2^{zero+} misses with respect to ORM2: value constraints; cardinality constraints; ring constraints; constraints among roles corresponding to non unary or non-total inclusion dependencies. There are strong computational reasons for not including these constraints. We plan to add to ORM2^{zero} both derivation rules and deontic constraints, in order to support sound, complete, and efficient reasoning for a significant fragment of ORM2.

In an earlier empirical study of conceptual models created at LogicBlox Inc. [6], the authors found that a restricted subset of ORM2, called ORM^{-}, includes the vast majority of constraints used in practice and, moreover, allows scalable test data generation. More recently [7], ORM^{-} has been extended to include features that resolved to be routinely used by LogicBlox developers. This last version of ORM^{-} includes: (i) simple mandatory constraints, (ii) frequency constraints (with the restriction that such a constraint spans one or more roles from the same fact type and the sets of roles spanned by different frequency constraints do not overlap), (iii) internal uniqueness constraints, (iv) subtype constraints, (v) value constraints, (vi) objectification, and (vii) external uniqueness constraints (with the restriction that one role in each of the involved fact types is not covered, and the types of the role players of the uncovered roles are 'type compatible'). Although an NP-hard algorithm for consistency checking is provided in [7], the algorithm itself is declared to be *not* complete under the ORM2 semantics (i.e. a full legal instance of an ORM^{-} schema may exist under the ORM2 semantics, even if the proposed algorithm has no solutions).

On the other hand, our paper introduced a *minimal* conceptual modelling language, called ORM2^{zero}, whose expressive power can be extended so as to cover the majority of the ORM2 constructs, and for which a complete algorithm for reasoning exists. We refer to the maximal expressivity covered by ORM2^{zero} as ORM2^{zero+}. ORM2^{zero+} has a clearly specified formal semantics and provably correct encoding into OWL2 ontology language, and its expressivity is comparable to that of ORM^{-}. Differently from ORM^{-}, our language has no special support for value constraints, since together with objectification falsify relation-descriptiveness. Nonetheless, ORM2^{zero} is able to represent many constraints which are not present in ORM^{-}. As for the frequency occurrences and the uniqueness, the restrictions imposed in ORM^{-} are similar to those we have in ORM2^{zero}.

Several extensions of ORM2^{zero} are currently under investigation. Among these, the integration of deontic modalities, allowing for the representation of obligation and permission rules, have been already deeply investigated in the context of the SBVR

language [8], and the results we got there could be easily transferred in ORM2^{zero}. Nonetheless, the most interesting extension of ORM2^{zero} is about the integration in the language of the so-called 'derivation rules': logical rules that may be used to derive new facts from other facts. Our main goal here is to identify a language for the specification of the rules that is decidable and whose computational properties do not negatively affect the complexity of the current ORM2^{zero}. Similar analyses in the context of conceptual modelling have been recently conducted for the combination of the UML class diagram and the rule language OCL (see, for example, [9]).

We thank two anonymous reviewers who helped us to fix several mistakes of a previous version of this papers.

References

1. Franconi, E., Mosca, A., Solomakhin, D.: ORM2: Formalisation and encoding in OWL2. In: Herrero, P., Panetto, H., Meersman, R., Dillon, T. (eds.) OTM-WS 2012. LNCS, vol. 7567, pp. 368–378. Springer, Heidelberg (2012)
2. Franconi, E., Mosca, A.: The formalisation of ORM2 and its encoding in OWL2. Technical Report KRDB12-2, KRDB Research Centre, Free University of Bozen-Bolzano (2012), `http://www.inf.unibz.it/krdb/pub/TR/KRDB12-2.pdf`
3. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The description logic handbook: theory, implementation, and applications. Cambridge University Press, New York (2003)
4. Halpin, T., Morgan, T.: Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design, 2nd edn. Morgan Kaufmann (2008)
5. Calvanese, D., Lenzerini, M., Nardi, D.: Unifying class-based representation formalisms. J. Artif. Intell. Res. (JAIR) 11, 199–240 (1999)
6. Smaragdakis, Y., Csallner, C., Subramanian, R.: Scalable satisfiability checking and test data generation from modeling diagrams. Automated Software Engineering 16(1), 73–99 (2009)
7. McGill, M.J., Dillon, L.K., Stirewalt, R.E.K.: Scalable analysis of conceptual data models. In: Proceedings of the 2011 International Symposium on Software Testing and Analysis, ISSTA 2011, pp. 56–66. ACM, New York (2011)
8. Franconi, E., Mosca, A., Solomakhin, D.: Logic-based reasoning support for SBVR. Fundamenta Informaticae 124, 1–18 (2013)
9. Queralt, A., Artale, A., Calvanese, D., Teniente, E.: OCL-lite: Finite reasoning on UML/OCL conceptual schemas. Data Knowl. Eng. 73, 1–22 (2012)