Supporting Conceptual Modelling in ORM by Reasoning

Francesco Sportelli(✉)

Free University of Bozen-Bolzano, Bolzano, Italy
fsportelli@unibz.it

Abstract. Object-Role Modelling (ORM) is a framework for modelling and querying information at the conceptual level. It comes to support the design of large-scale industrial applications allowing the users to easily model the domain. The expressiveness of the ORM constraints may lead to implicit consequences that can go undetected by the designer in complex diagrams during the software development life cycle. To avoid these issues we perform the reasoning on ORM diagrams in order to detect relevant formal properties, such as inconsistencies or redundancies, that cause a software quality degradation leading to an increment of development times and costs.

In this paper we present an extension of ORM formalisation by Derivation Rules, which are additional ORM constructs that capture some relevant information of the domain that cannot be expressed in standard ORM.

Moreover, we provide a tool (UCM Framework) which enables reasoning on conceptual modelling software along with an implemented case of study (ORMiE).

Keywords: ORM · Conceptual modelling · Reasoning · Rules

1 Overview

Conceptual modelling is a critical step during the development of a database system. It is the detailed description of the universe of discourse in a language that is understandable by users of the business domain. Object-Role modelling (ORM) is a conceptual language for modelling, which includes a graphical and textual language for specifying models, a textual language for formulating queries, as well as procedures for constructing ORM models, and mapping to other kinds of models like UML and ER. ORM is fact-oriented, i.e., it models the information in a way that it can be verbalized using sentences that are easily understandable by domain experts and even for those who are not familiar with IT in general. Unlike ER and UML, fact-oriented models are attribute-free, treating all facts (sentences) as relationships (unary, binary, ternary etc.) and this makes it more stable and adaptable to changing business requirements. For example, instead of using the attributes Person.isSmoker and Person.hiredate, fact-oriented models use the fact types Person smokes and Person was hired on Date [1].

M. Kirikova et al. (Eds.): ADBIS 2017, CCIS 767, pp. 422–431, 2017.
DOI: 10.1007/978-3-319-67162-8_41

The ORM constraints expressiveness may lead to implicit consequences that can go undetected by the designer in complex diagrams; this may also lead to various forms of inconsistencies or redundancies in the diagram itself that give rise to the degradation of the quality of the design and/or increased development times and costs. The approach used to solve this issue involves the automated reasoning in order to detect inconsistencies and redundancies. Moreover, ORM diagrams can be equipped with Derivation Rules which are additional ORM constructs which are able to express knowledge that is not expressible with standard ORM. The usage of those rules brings to a further complexity so the goal of this paper is to detect a decidable fragment in order to perform the reasoning task even on those ORM diagrams equipped with those rules.

We also introduce the state of the art starting from the first ORM formalisation until the most recent one. Then we introduce an overview of the ORM language, focusing on its main features like fact-oriented, the verbalisation and the graphical notation, providing also the running example that will be shown in the rest of the paper. The section concerning the research has three subsections:

1. UCM Framework, a tool designed to activate automated reasoning on conceptual modelling software;
2. ORMiE, a tool which performs reasoning on ORM diagrams using UCM Framework;
3. Derivation Rules, where we show the results achieved so far concerning the formalisation.

We conclude the paper presenting the list of what is still to be done in the frame of the PhD.

2 Motivation

The ORM formalisation has been treated in years of research and still today it is a topic of interest. Formalising such language allows to activate reasoning procedures, carried out by Description Logics reasoners. Since the reasoning is able to detect relevant formal properties, such as inconsistencies or redundancies, the purpose of the reasoning is to support the modeller during the modelling which is a delicate process during the development of a software or a database. Without this support such issues could lead to a degradation of the quality of the design and an increase of development times and costs. Especially in large diagrams those issues are hard to spot by naked eye, so the need of this approach is crucial to prevent mistakes during the software or database development.

Although several papers presented their own ORM formalisation, no one has taken into account the formalization of derivation rules so far. Derivation rules are new ORM constraints which are able to express knowledge that is beyond normal ORM capabilities, but this feature leads to an increase of expressiveness of the diagrams. For this reason, the challenge is to identify a decidable fragment in order to extend the reasoning even on those ORM diagrams equipped with those rules.

Another challenge has more a methodological flavour, which involves the development of a system that is able to extend any conceptual modelling applications with reasoning services. The direct impact of this research involves the modellers, the developers and those who need a support tool which easily checks the consistency of conceptual schema in order to save time during the development life cycle.

3 Related Work

The ORM formalisation started with Terry Halpin's PhD Thesis [2]. In the context of design conceptual and relational schemas, Halpin formalized the NIAM language that is the ancestor of ORM. In his thesis there is the first attempt to formalize a modelling language in order to perform the reasoning task, so the main objective is to provide formal basis for reasoning about conceptual schemas and for making decision choices. After the spreading of ORM and its implementation in NORMA [3,4], ORM became more popular so the logicians' community took into account the possibility to formalize this very expressive language.

In 2007, Jarrar formalizes ORM using $\mathcal{DLR}_{ifd}$ [5], an extension of Description Logics introduced in [6]. The paper shows that a formalisation in OWL $\mathcal{SHOIN}$ would be less efficient than $\mathcal{DLR}_{ifd}$ because some ORM constraints cannot be translated (predicate uniqueness, external uniqueness, set-comparison constraints between single roles and between not contiguous roles, objectification n-ary relationships). In [7], Jarrar encodes ORM into OWL $\mathcal{SHOIN}$. Another formalisation of ORM in $\mathcal{DLR}_{ifd}$ was done by Keet in [8].

In 2009 OWL2 was recommended by W3C Consortium as a standard of ontology representation on the Web bringing some benefits: it is the recommended ontology web language; it is used to publish and share ontologies on the Web semantically; it is used to construct a structure to share information standards for both human and machine consumption; automatic reasoning can be done against ontologies represented in OWL2 to check consistency and coherency of these ontologies.

An ORM formalisation based on OWL2 is proposed by Franconi in [9], where he introduces a new linear syntax and FOL semantics for a generalization of ORM2, called ORM2plus, allowing the specification of join paths over an arbitrary number of relations. The paper also identifies a "core" fragment of ORM2, called ORM2zero, that can be translated in a sound and complete way into the ExpTime-complete Description Logic $\mathcal{ALCQI}$. In [10] is provided a provably correct encoding of a fragment of ORM2zero into a decidable fragment of OWL2 and it is discussed how to extend ORM2zero in a maximal way by retaining at the same time the nice computational properties of ORM2zero.

The most recent paper related to ORM formalisation is [11] where Artale introduces a new extension of $\mathcal{DLR}$, namely $\mathcal{DLR}^{+}$. This paper is strictly connected with this work because the logic $\mathcal{DLR}^{+}$ it is meant to represent n-ary

relationships which are suitable for languages like ORM. The ORM implementation we use is an ongoing work based on $\mathcal{DLR}^{+}$. In particular, the decidable fragment we use is $\mathcal{DLR}^{\pm}$, obtained by imposing a simple syntactic condition on the appearance of projections and functional dependencies in $\mathcal{DLR}^{+}$. In the paper is also provided an OWL encoding and it is proved that $\mathcal{DLR}^{\pm}$ captures a significant fragment of ORM2.

Since this work is also focused on the formalisation of derivation rules, we need to mention OCL. OCL stands for Object Constraint Language, it is the declarative language for describing rules that apply to UML diagrams for defining constraints in order to support the conceptual modelling, like Derivation Rules for ORM. In [12] has been provided a formalisation of a fragment of this language and has been also proved the equivalence between relational algebra and the fragment with only FOL features, namely OCL_{FO}.

We conclude this section stating that, to best of our knowledge, we use $\mathcal{DLR}^{\pm}$ in order to build a decidable ORM mapping into OWL. In particular, we focus on Derivation Rules formalisation.

4 ORM

ORM stands for Object-Role Modelling. It is a language that allows users to model and query information at the conceptual level where the world is described in terms of *objects* (things) playing *roles* (parts in relationships) [13]. The idea behind ORM and its approach is that an object-role model avoids the need to write long documents in ambiguous natural language prose. It's easy for non-technical sponsors to validate an object-role model because ORM tools can generate easy-to-understand sentences. After an object-role model has been validated by non-technical domain experts, the model can be used to generate a class model or a fully normalised database schema. ORM main features are:

- *fact-oriented*, all facts and rules are modelled in terms of controlled natural language (FORML) sentences easy to understand even for non-technical users;
- *attribute-free*, unlike ER and UML, makes it more stable and adaptable to changing business requirements;
- *graphical*, it has a graphical notation implemented by the software NORMA;
- *formalised*, it has a clear syntax and semantics, so reasoning on an ORM diagram is enabled.

Unlike ER or UML, ORM makes no use of attributes in its base models; although this often leads to larger diagrams, an attribute-free approach has advantages for conceptual analysis, including simplicity, stability, and ease of validation. Attribute-free models with a controlled natural language facilitate model validation by verbalisation and population. Model validation should be a collaborative process between the modeller and the business domain expert who best understands the business domain. All facts, fact types, constraints and derivation rules may be verbalised naturally in unambiguous language that is

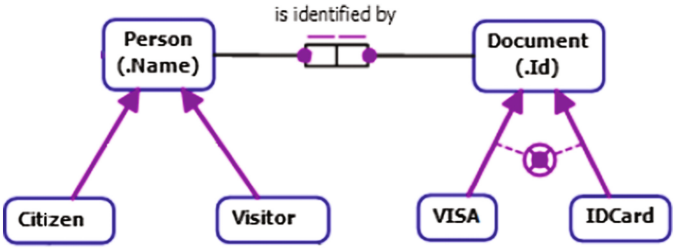

Fig. 1. ORM diagram example

easily understood by domain experts who might not be experts in the software systems ultimately used for the physical implementation.

The meaning of the diagram in Fig. 1 is the following: a person can be a citizen or a visitor; each person is identified by one and only one document which can only be either a visa or an id card. The entities are depicted by smooth rectangles and the relationships by a sequence of tiny boxes according to the cardinality relationship. The purple dot represents the mandatory constraint, the dash on the tiny rectangle box is the uniqueness constraint, the equivalent of the relational keys. The arrows among entities represents the ISA relationship. Finally, the circle with the cross inside means disjointness; the one with another circle inside means covering; the combination of this two is the circle we see between Visa and IDCard. The notation (.Name) and (.Id) inside Person and Document it is a graphical shortcut provided by NORMA for top level entities. Intuitively, it means that each person has a name and each document has an id. The corresponding FORML verbalization is the following:

```
Person is an entity type.
Citizen is an entity type.
Visitor is an entity type.
Document is an entity type.
VISA is an entity type.
IDCard is an entity type.
Person is identified by Document.
Each Citizen is an instance of Person.
Each Visitor is an instance of Person.
Each VISA is an instance of Document.
Each IDCard is an instance of Document.
Each VISA is an instance of Document.
Each IDCard is an instance of Document.
Each Person has exactly one Document.
For each Document, at most one Person has that Document.
For each Document, exactly one of the following holds: that Document is
some VISA; that Document is some IDCard.
```

This feature turns out to be helpful during the modelling phase especially when the non-IT stakeholders interact with the software engineers in order to reach a mutual comprehension about the meaning of the diagram. For example, if the non-IT stakeholder

detects unexpected sentences which do not reflect the software specifications, it is easy for the modeller to modify the interested part.

5 Contributions

The main goal of the research concerns to enrich the conceptual modelling by reasoning in order to detect constraints which can lead to unexpected software behaviours, or to infer new knowledge. This research is characterized by the synergy of the methodological and the theoretical aspects. UCM Framework (Universal Conceptual Modelling Framework) and ORMiE (ORM Inference Engine) are tools developed to implement the ORM language in order to perform the reasoning over ORM schemas. Moreover, the theoretical part is focused on the formalisation of ORM Derivation Rules which has been also implemented in the aforementioned tools.

5.1 UCM Framework

Usually conceptual modelling tools do not take into account the problem of checking whether the semantics of the conceptual schema is consistent or not. To tackle this situation we developed UCM Framework which activates reasoning on conceptual modelling applications. UCM Framework has several features: it provides API for developers, reasoning services, the import/export of ontologies and diagrams in different languages like ORM, UML and ER. In Fig. 2 is shown the architecture of the framework. Each conceptual modelling application communicates with the core system using a specific driver, both for input and for the output where the inferences are encoded. The input schema is first encoded in a data structure (UCM Model) by API services, then the reasoning is performed by Fact++ reasoner [14]. After that, the inferences are stored into another data structure (UCM Inferred Model) by API and inferences are delivered to the destination application by drivers. Using this approach one can easily integrate this framework in order to enrich its conceptual modelling application by reasoning. Currently, two applications use this framework: Menthor [15] and ORMiE.

5.2 ORMiE

ORMiE (ORM Inference Engine) is an extension of NORMA, which is the official ORM-based Microsoft Visual Studio conceptual modelling tool [3]. ORMiE uses UCM Framework, so it is just one example how UCM Framework works on a target ORM-based software. ORMiE activates automated reasoning over ORM diagrams providing an interface where mistakes, redundancies or more in general new inferred knowledge are shown. It takes advantage of all nice features from Visual Studio Framework being such a powerful tool for those who need to model a domain following the ORM methodology. Moreover, ORMiE is able to perform the reasoning on those ORM diagrams equipped with Derivation Rules.

5.3 Derivation Rules

Derivation Rules are special ORM constructs which express knowledge that would otherwise not be expressible by standard ORM, so their expressibility is far reaching than

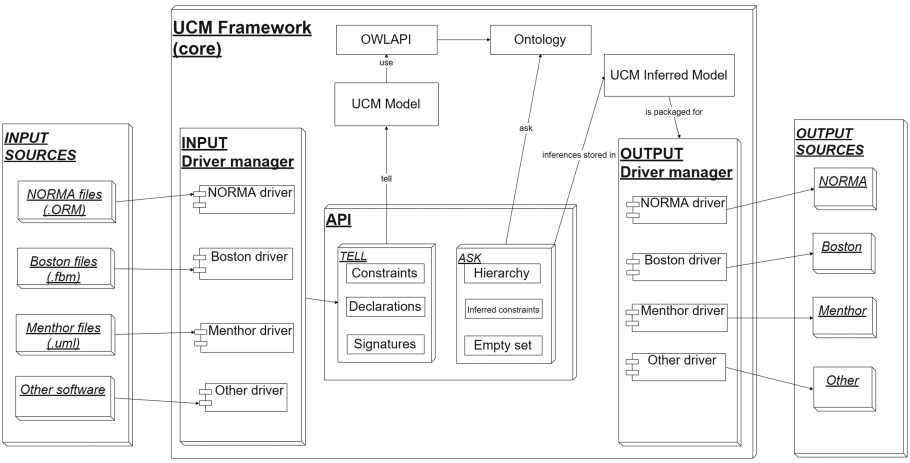

Fig. 2. UCM framework architecture

standard ORM. Their purpose is to derive new information from other information, like triggers, stored procedures and views in SQL. The goal is to enable the reasoning on those rules in order to extend the reasoning even on those ORM diagrams equipped with Derivation Rules. There are two kind of Derivation Rules: the Subtype Derivation Rules and the Fact Type Derivation Rules. A Subtype Derivation Rule defines all the instances which belongs to a subentity by a set of constraints defined in the rule definition. The reason because those rules are applied on subentities is because in some diagrams the is-a relationship between entities is too weak to capture the entire desired semantics of the diagram; a FactType Derivation Rule is placed on the predicates, namely the ORM roles.

The Derivation Rules we are focusing on are Subtype Derivation Rules. We want to formalise those rules in order to activate the reasoning on those diagrams with Subtype Derivation Rules. To achieve this goal it is important to take into account that the reasoning lays on logic, so we need to find a way to encode derivation rules into a logical language. First of all we need to understand how a derivation rule is made from a structural point of view, in other words we need to detect a clear syntax. Then, at the syntax is assigned a corresponding semantics and in the end an encoding into a logical language is performed in order to made the reasoning possible. In [16] is provided the full methodology used to formalise those rules and their mapping into OWL.

We provide an example with the graphical notation implemented by NORMA. For example, the diagram in Fig. 1 does not tell us which are *exactly* the people in the entity Citizen and *exactly* the people in the entity Visitor. We only know that a person can be a citizen or a visitor, but in the ORM standard notation there are no constraints able to capture these sets. If we want to use this knowledge we have to use the ORM Derivation Rules like in Fig. 3.

As we can see, a derivation rule is defined by an asterisk on entities and a text which defines the meaning of the rule. We state that all the people which are identified by an id card are citizens; all the people which are identified by a visa are visitors. It

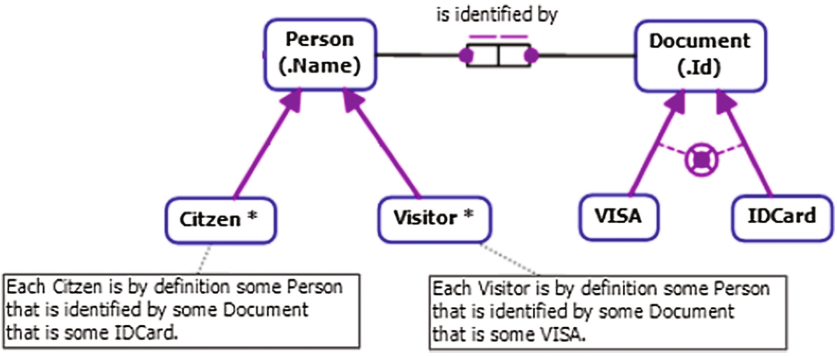

Fig. 3. ORM diagram example with derivation rules

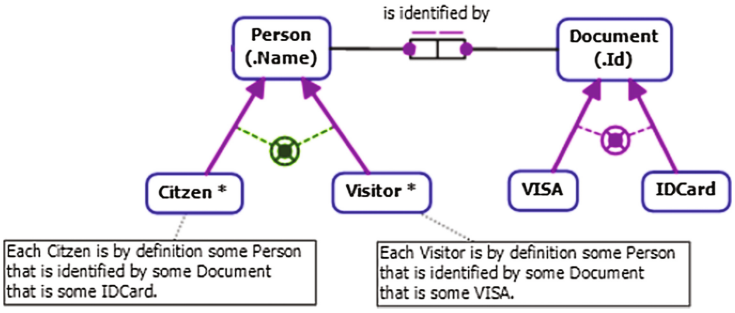

Fig. 4. Inferred disjoint and covering constraints

is important to observe that the text is not just a collections of words, instead it is in controlled-natural language format that is to say it is well defined by a precise syntax.

What can be the outcome of the diagram in Fig. 3? The answers is in Fig. 4. We obtained a disjunction and covering between the entities Citizen and Visitor. The disjunction is inferred because there is no chance to find a common element between the entity Visa and IDCard. Since the derivation rules capture separately the two sets, visitors and citizens, even the corresponding entities have no element in common. What about the covering? Since Visa and IDCard cover Document and since Person has the mandatory constraint on the relationship *is identified by*, each person must participate to this relation; in addition to this, the two derivation rules ensure that the sum of the instances in Citizen and Visitor are exactly those who are in Person. So it is not possible to find an instance which is not in Citizen of in Visitor. To prove this, now we add the entity Illegal: people without documents, neither a visa nor id card. The outcome of the reasoning is shown in Fig. 5. Illegal is red because it is an empty set. This means that the only consistent world is given by the entity Illegal with no instances, because there are no instances in Illegal which satisfy the rules of the diagram. Again, this is because Person has the mandatory constraint on the *is identified by* relationship and because the set of Person is already taken by the entities Citizen and Visitor. Therefore, there

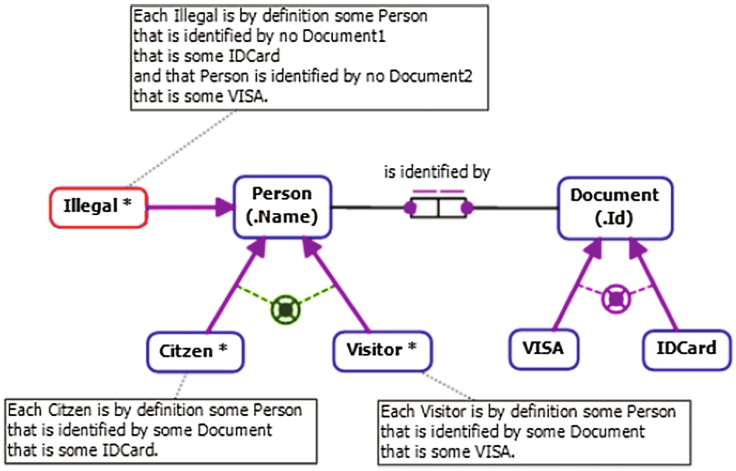

Fig. 5. Illegal is inconsistent

is no way an instance in Illegal could be in Person. The counter-example is trivial: if we remove the mandatory constraint on Person then Illegal would not be inconsistent anymore.

6 Conclusion and Future Works

We have seen in this paper an extension of the current ORM formalisation introducing ORM Derivation Rules. These rules express knowledge that is beyond ORM capabilities, but they are far expressive than standard ORM. Therefore, we have formalised a non-trivial decidable fragment in order to enable the reasoning over those ORM diagrams equipped with these rules. The reasoning procedure detects relevant formal properties as inconsistencies, redundancies or implicit constructs, helping the modeller to prevent unexpected software behaviours.

We presented UCM Framework, a tool which is specifically designed to enrich with reasoning any conceptual modelling software. It supports popular conceptual modelling languages like ER, UML and even ORM. A tool which makes use of this framework is ORMiE, a Microsoft Visual Studio plugin used to manage ORM diagrams.

The research and development of UCM Framework continues on two tracks: from one side we plan to add the explanation feature in order to enhance the understanding of the diagram by the user perspective, since this service explains why and how something went wrong during the modelling; while on the other hand we plan to extend the reasoning even on the instances of the conceptual schema.

Finally, an ongoing theoretical work concerns the formalisation of Fact Type Derivation Rules in order to capture a relevant decidable fragment that will be implemented in the UCM Framework.

References

1. Halpin, T.A.: Object-role modeling: Principles and benefits. IJISMD **1**(1), 33–57 (2010)
2. Halpin, T.: A Logical Analysis of Information Systems: static aspects of the data-oriented perspective. PhD thesis (July 1989)
3. Curland, M., Halpin, T.: The NORMA software tool for ORM 2. In: Soffer, P., Proper, E. (eds.) CAiSE Forum 2010. LNBIP, vol. 72, pp. 190–204. Springer, Heidelberg (2011). doi:10.1007/978-3-642-17722-4_14
4. Sportelli, F.: NORMA: A software for intelligent conceptual modeling. In: Proceedings of the Joint Ontology Workshops 2016 Episode 2: The French Summer of Ontology co-located with the 9th International Conference on Formal Ontology in Information Systems (FOIS 2016), Annecy, France, 6–9 July 2016 (2016)
5. Jarrar, M.: Towards automated reasoning on ORM schemes. In: 26th International Conference on Conceptual Modeling, ER 2007, pp. 181–197 (2007)
6. Calvanese, D., De Giacomo, G., Lenzerini, M.: Identification constraints and functional dependencies in description logics. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, 4–10 August 2001, pp. 155–160 (2001)
7. Jarrar, M.: Mapping ORM into the SHOIN/OWL description logic. In: On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops, OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, OTM Academy Doctoral Consortium, MONET, OnToContent, ORM, PerSys, PPN, RDDS, SSWS, and SWWS 2007, Proceedings, Vilamoura, Portugal, 25–30 November 2007, Part I, pp. 729–741 (2007)
8. Keet, C.M.: Mapping the object-role modeling language ORM2 into description logic language dlrifd. CoRR, abs/cs/0702089 (2007)
9. Franconi, E., Mosca, A., Solomakhin, D.: The formalization of ORM2 and its encoding in OWL2. In: International Workshop on Fact-Oriented Modeling (ORM 2012) (2012)
10. Franconi, E., Mosca, A.: Towards a Core ORM2 language (Research Note). In: Demey, Y.T., Panetto, H. (eds.) OTM 2013. LNCS, vol. 8186, pp. 448–456. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41033-8_58
11. Artale, A., Franconi, E.: Extending DLR with labelled tuples, projections, functional dependencies and objectification. In: Proceedings of the 29th International Workshop on Description Logics (2016)
12. Franconi, E., Mosca, A., Oriol, X., Rull, G., Teniente, E.: Logic foundations of the OCL modelling language. In: Fermé, E., Leite, J. (eds.) JELIA 2014. LNCS, vol. 8761, pp. 657–664. Springer, Cham (2014). doi:10.1007/978-3-319-11558-0_49
13. Halpin, T.A., Morgan, T.: Information Modeling and Relational Databases, 2nd edn. Morgan Kaufmann, San Francisco (2008)
14. Fact++ reasoner. http://owl.man.ac.uk/factplusplus/
15. Moreira, J.L.R., Sales, T.P., Guerson, J., Braga, B.F.B., Brasileiro, F., Sobral, V., Menthor editor: An ontology-driven conceptual modeling platform. In: Proceedings of the Joint Ontology Workshops 2016 Episode 2: The French Summer of Ontology co-located with the 9th International Conference on Formal Ontology in Information Systems (FOIS 2016), Annecy, France, 6–9 July 2016 (2016)
16. Sportelli, F., Franconi, E.: Formalisation of ORM derivation rules and their mapping into OWL," in On the Move to Meaningful Internet Systems: OTM 2016 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2016, Proceedings, Rhodes, Greece, 24–28 October 2016, pp. 827–843 (2016)