NORMA: a Software for Intelligent Conceptual Modeling

Francesco Sportelli
KRDB, Department of Computer Science, Free University of Bozen-Bolzano, Italy
francesco.sportelli@inf.unibz.it

Abstract. Object-Role Modeling (ORM) is a framework for modeling and querying information at the conceptual level. It comes to support the design of large-scale industrial applications allowing the users to model easily the domain. The reasoning on a conceptual schema enables to automatically detect relevant formal properties, such as inconsistencies or redundancies that cause a degradation of the quality of the design and an increase of development times and costs. In this demonstration we introduce NORMA, a tool which implements the ORM language and its plugin ORMIe, which performs the reasoning on ORM conceptual schema in order to help the modeler during the modeling phase to avoid mistakes that could lead the software to unexpected behaviors.

Keywords. Conceptual modeling, OWL, Reasoning, Description Logics, ORM

1. Introduction

Conceptual modeling 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 Modeling (ORM) is a conceptual language for modeling, which includes a graphical (ORM's graphical notation) and textual language (FORML) 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. The expressiveness of the ORM constructs may lead to implicit consequences that can go undetected by the designer in complex schemas; 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. This issue leads to the need of automated reasoning to check the mentioned inconsistencies and redundancies. NORMA is the tool that implements the ORM language and ORMIe is a plugin for NORMA which enables the automated reasoning on ORM conceptual schemas. Of particular interest are the Derivation Rules, they are special ORM constructs which are able to express knowledge that is beyond the ORM capabilities. The current version of ORMIe is able to perform reasoning on a subset of such Derivation Rules.

2. ORM and NORMA

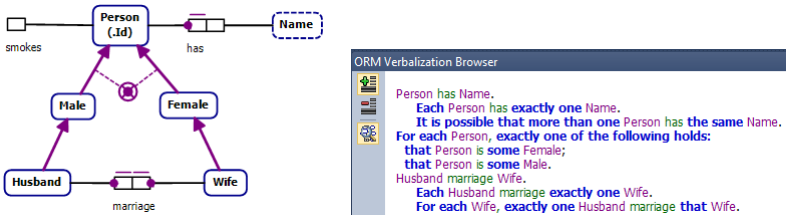
ORM stands for Object-Role Modeling. It is a language that allows users to model and query information at the conceptual level and the world is described in terms of *objects* (things) playing *roles* (parts in relationships) [1], [2]. This language has been formalized in Terry Halpin's PhD Thesis [3] in the context of conceptual modeling. During last years, the logicians' community has focused the attention on the formalization of ORM into a logical language, precisely Description Logics [4], [5], [6], [7]. Recently, ORM has evolved in ORM2 which has been formalized in [8]. The formalization of ORM allows to perform automated reasoning in order to detect relevant formal properties and new inferred knowledge in a given ORM diagram.
Two software that implement the reasoning are ICOM [9] and DogmaModeler [10]. ICOM is an advanced conceptual modeling tool that allows the user to design multiple ER or UML class diagrams with inter and intra-model constraints, expressed in a rich view language similar to OCL and relational algebra based on Description Logics. DogmaModeler is an ontology modeling tool that implements ORM supporting ontology modularization and composition and verbalization into pseudo natural languages.

NORMA is the tool that implements the ORM language, it has been developed by Terry Halpin and Matthew Curland, [11], [12]. NORMA stands for *Natural Object-Role Modeling Architect*, it is a free software available from the Sourceforge repository and it is implemented in Microsoft Visual Studio Environment and it is written in C#. Unlike ICOM and DogmaModeler, NORMA is built into this environment so it inherits all the functionalities of the popular framework and it takes advantage of full API support that makes the software easily to maintain; moreover, it has a plugin-like structure which makes possible for developers to create their own plugins to extend NORMA. NORMA implements the ORM's graphical notation for all ORM constraints making easier for the final user to model a conceptual schema. From the reasoning perspective, unlike ICOM and DogmaModeler, NORMA is currently able to cover reasoning even on Derivation Rules (see below). An exhaustive demo of NORMA features can be found here: *https://vimeo.com/159092232* .

ORM main features are:

- *formalized*, it has a clear syntax and semantics;
- *graphical*, has an official software (NORMA) that makes possible to express the conceptual schema in an easy graphical way, like the examples shown in this paper;
- *fact-oriented*, all facts and rules are modeled in terms of controlled natural language (FORML), sentences easy to understand even for non-technical users;
- *attribute-free*, unlike ER and UML, it is more stable and adaptable to changing business requirements.

The following examples show how NORMA works and implements ORM. We show only the very basic ORM constructs to simplify the general understanding. In **Figure 1a** we have an ORM conceptual schema with some entities: *Person*, *Male*, *Female*, *Husband* and *Wife*. *Male* and *Female* are subentities of *Person* in disjoint and covering. *Husband* is subentity of *Male* and *Wife* is subentity of *Female*. *Name* is an entity value which means a collection of data types (string, int, etc.). The relations are depicted with the tiny rectangle boxes. The arity depends on the number of boxes (*roles*) and consequently of



(a) *ORM schema.* (b) *ORM schema verbalization.*

Figure 1. The ORM graphical language in NORMA

role players (*entities*). *Smokes* is a unary relation that is exactly like the boolean value in a SQL column true/false. *Has* and *marriage* instead are binary relations. The fact-oriented approach facilitates the understanding of the ORM conceptual schema in a way that even the non-technical users can easily understand the semantics of the schema, since it is expressed in controlled natural language; this approach helps also the modeler during the validation of the schema to detect errors. This feature is implemented in NORMA by ORM Verbalization Browser, as shown in **Figure 1b**.

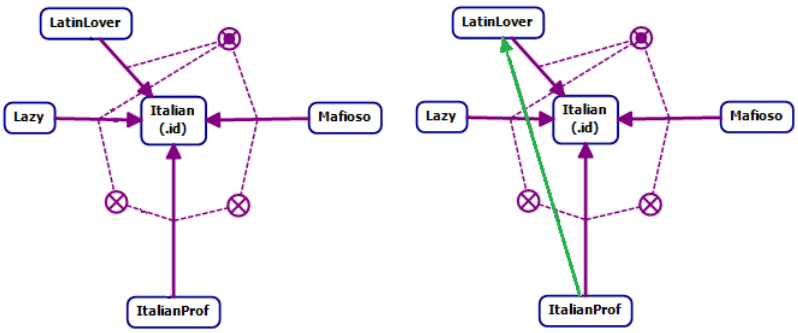
3. ORMIe Plugin

ORMIe stands for *ORM Inference Engine* and it is a NORMA plugin that performs reasoning on ORM schemas. Reasoning helps the modeler to detect relevant formal properties of the schema that may be undetected during the modeling phase which give rise to the software quality degradation and/or increased development times and costs. Especially with large software this is helpful to understand if and why the schema could have some inconsistencies, redundancies and unexpected behaviors. The reasoning services in ORM are due to off-the-shelves reasoners developed by the Description Logics community. As in [13] for UML, we have a polynomial encoding of a subset of ORM language in the Description Logics $\mathcal{ALCQI}$ and this encoding preserves enough semantics to keep reasoning on ORM sound and complete. Once ORMIe is activated, it starts its inference engine translating the ORM conceptual schema in OWL generating an ontology; after that, the ontology is processed by Fact++ [14] reasoner in order to discover the implicit knowledge, like inconsistencies or redundancies.

In **Figure 2a** we can see an example on how this can help the modeler during the modeling phase. The ORM conceptual schema contains the entity Italian and its subentities: *LatinLover*, *Lazy* and *Mafioso*, that are in covering and disjoint. Then, we also have another subentity called *ItalianProf*, which is in disjointed with *Lazy* and *Mafioso*. Considering that the covering and the disjointness of the *Italian* set is made by *LatinLover*, *Lazy*, *Mafioso* and that *ItalianProf* cannot be *Lazy* and *Mafioso*, we can conclude that all the objects in *ItalianProf* must also be objects of *LatinLover*. The result of the reasoning is shown in **Figure 2b**.

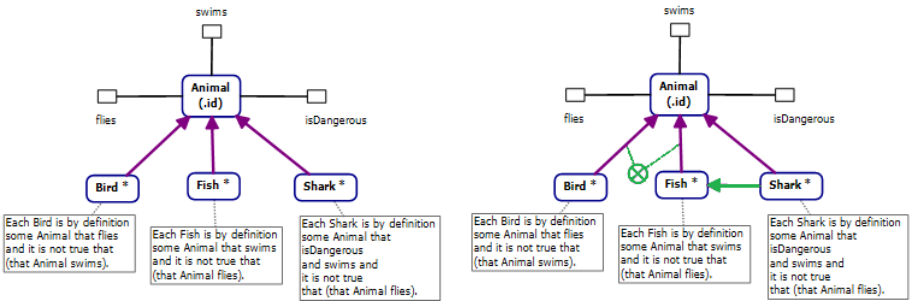
4. Reasoning on Derivation Rules

ORM is enriched with Derivation Rules that are of particular interest for our demonstration, because they are able to express knowledge that is beyond ORM capabilities and



(a) *ORM conceptual schema without reasoning*. (b) *ORM conceptual schema with reasoning*.

Figure 2. Reasoning over ORM conceptual schema.



(a) *Derivation Rules without reasoning*. (b) *Derivation Rules with reasoning*.

Figure 3. Reasoning on Derivation Rules

to derive new information from other information, similar to triggers, stored procedures and views in SQL. In NORMA these rules are expressed in FORML, which is a controlled natural language. This means that Derivation Rules are well structured and they have a precise syntax. This syntax has been studied and enriched with an unambiguous semantics, so it was possible to catch the encoding in $\mathcal{ALCQI}$ and consequently in OWL.

In **Figure 3a** there is an example of an ORM conceptual schema with 3 Derivation Rules (depicted with an asterisk in the entities and a textbox):

- Bird is an Animal that flies and doesn't swim;
- Fish is an Animal that swims and doesn't fly;
- Shark is an Animal that swims, doesn't fly and is dangerous.

Since all the objects in *Bird* fly and never swim and vice versa for those that are in *Fish*, it is pretty clear that a bird cannot be a fish and vice versa, so this constraint leads to the disjointness between the corresponding entities. The entity *Shark* is a specialization of the entity *Fish*, because it has the same properties of *Fish*, but it is also *"dangerous"*. This leads to the inferred constraints in **Figure 3b**.

Derivation Rules can be placed on subtype entities and also on relations. Until now, ORMIe covers reasoning only for Subtype Derivation Rules.

5. Conclusions and Future Works

We presented ORM, a modeling language used to design diagram at the conceptual level. ORM is implemented in NORMA, an advanced conceptual modeling tool that extended with the plugin ORMIe is able to perform automated reasoning to support the conceptual modeling phase during the development of the software lifecycle. We demonstrated, by means of examples, the importance of the reasoning since it provides implicit information which can be undetected during the modeling phase. This approach helps the modeler to preserve the quality of the conceptual schema and to avoid mistakes that could lead the software to unexpected behaviors.

The research and development of NORMA continues on two tracks: we are keeping studying the formalization of Derivation Rules in order to cover binary relations as well; we are currently considering to implement the explanation service, which could help the modeler to understand the reason behind mistakes that may occur during the modeling phase.

References

[1] T. A. Halpin. Object-role modeling: Principles and benefits. *IJISMD*, 1(1):33–57, 2010.

[2] T. A. Halpin and T. Morgan. *Information modeling and relational databases (2. ed.)*. Morgan Kaufmann, 2008.

[3] T. Halpin. *A Logical Analysis of Information Systems: static aspects of the data-oriented perspective*. PhD thesis, jul 1989.

[4] C. M. Keet. Mapping the object-role modeling language ORM2 into description logic language dlrifd. *CoRR*, abs/cs/0702089, 2007.

[5] M. Jarrar. Towards automated reasoning on ORM schemes. In *Conceptual Modeling - ER 2007, 26th International Conference on Conceptual Modeling, Auckland, New Zealand, November 5-9, 2007, Proceedings*, pages 181–197, 2007.

[6] R. Hodrob and M. Jarrar. Mapping ORM into OWL 2. In *Proceedings of the 1st International Conference on Intelligent Semantic Web-Services and Applications, ISWSA 2010, Amman, Jordan, June 14-16, 2010*, page 9, 2010.

[7] M. Jarrar. 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, Vilamoura, Portugal, November 25-30, 2007, Proceedings, Part I*, pages 729–741, 2007.

[8] E. Franconi and A. Mosca. Towards a core ORM2 language (research note). In *On the Move to Meaningful Internet Systems: OTM 2013 Workshops - Confederated International Workshops: OTM Academy, OTM Industry Case Studies Program, ACM, EI2N, ISDE, META4eS, ORM, SeDeS, SINCOM, SMS, and SOMOCO 2013, Graz, Austria, September 9 - 13, 2013, Proceedings*, pages 448–456, 2013.

[9] P. R. Fillottrani, E. Franconi, and S. Tessaris. The ICOM 3.0 intelligent conceptual modelling tool and methodology. *Semantic Web*, 3(3):293–306, 2012.

[10] Dogmamodeler. `https://sourceforge.net/projects/dogmamodeler/`.

[11] Orm on sourceforge. `https://sourceforge.net/projects/orm/`.

[12] The orm foundation. `https://www.ormfoundation.org`.

[13] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artif. Intell.*, 168(1-2):70–118, 2005.

[14] Fact++ reasoner. `http://owl.man.ac.uk/factplusplus/`.