Prospects for and issues with mapping the Object-Role Modeling language into $\mathcal{DLR}_{ifd}$

C. Maria Keet

Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
keet@inf.unibz.it

Abstract. Object-Role modellers miss the advantages of automated reasoning over their ORM conceptual models, which could be addressed by DL reasoners. DLs are not considered user-friendly and could benefit from the easy to use ORM diagrammatic and verbalization interfaces and modelling methodologies. Relating the two would greatly expand the scope for automated reasoning with additional scenarios to improve quality of software systems. Given that none of the extant DL languages are as expressive as ORM or its successor ORM2, the 'best-fit' $\mathcal{DLR}_{ifd}$ was chosen to map the formal conceptual modelling language ORM2. For the non-mappable constraints, pointers to other DL languages are provided, which could serve as impetus for research into DL language extensions or interoperability between existing DL languages.

1 Introduction

Description Logic (DL) languages have been shown useful for reasoning both over conceptual models like ER and UML [2, 4, 12] and ontology languages such as OWL-DL, OWL-Lite [27], its proposed successor OWL 1.1 [26] that is based on the DL language $\mathcal{SROIQ}$ [22], and *DL-Lite* [7]. In particular, we are interested in the notion of using DLs as unifying paradigm for conceptual modelling to enable automated reasoning over conceptual data models, which, be it due to legacy, preference, or applicability, are made with different conceptual modelling languages. A tool such as iCOM [15] already supports automated reasoning over UML or EER diagrams, which may have cross-conceptual model assertions. What is lacking, however, is a mapping from Object-Role Modeling (ORM) into a DL. One may wonder: why yet another mapping? There are three main reasons for this. First, ORM is a so-called "true" conceptual modelling language in the sense that it is independent of the application scenario and it has been mapped into both UML class diagrams and ER. That is, ORM and its successor ORM2 can be used in the conceptual analysis stage for database development, application software development, requirements engineering, business rules, and other areas, *e.g.*, [3, 5, 14, 17, 24]. Thus, if there is an ORM-to-DL mapping, the possible applications for automated reasoning services can be greatly expanded. Second, an important aspect of ORMing is to have great consideration for the user and therefore ORM tools such as CaseTalk and NORMA are very user-friendly, so that even domain experts unfamiliar with formalisms can start modelling after

half an hour training. ORM tools have both diagrammatic and textual interfaces (the latter through so-called verbalizations, which are pseudo-natural language renderings of the axioms), thereby accommodating different user preferences toward modelling. Third, ORM is more expressive than either UML or EER and is more expressive than the extant DLs as well. Most ORM constraints are supported in one DL language or another, but none supports all ORM constraints. The proposed ORM-to-$\mathcal{DLR}_{ifd}$ mapping may provide some élan to examine DL language extensions not only based on interest and particular user requests from domain-modelling scenarios, but toward those (combinations of) extensions that are already known to be useful, or to find an implementable solution where for different (sections of) conceptual models, different languages can be used within one application interface.

The remainder of this paper is organised as follows. Subsections 1.1 and 1.2 contain brief introductions to ORM and $\mathcal{DLR}$, respectively. The main part is devoted to the assessment of the mapping (Section 2). Finally, some reflections and conclusions are included in Section 3.

1.1 Brief introduction to Object-Role Modeling (ORM)

The basic building blocks of the ORM language are object types, value types—at the conceptual level no subjective distinction has to be made between classes and attributes—relations, roles, and a plethora of constraints. A role is that what the object or value type 'plays' in the relation. ORM supports n-ary relations, where n is a finite integer ≥ 1. An example of a fact type is shown in Fig.1, which was made with the NORMA CASE tool [25]: the diagrammatic representation of the relation –rectangle divided into three roles, one for each participating object or value type– in the ORM model has 1) the name of the relation, which is displayed in the properties box of the relation and is generated automatically by the software (called "PatientAdmittedToHospitalAtDateDate" in the example), 2) role names, such as "[hospitalAdmission]" for the the role that object type Patient plays, and 3) a label attached to the relation, "... admitted to ... at date ...", which is used for the verbalization. ORM models can be mapped into, among others, ER and UML diagrams, IDEFX logical models, SQL table definitions, C, Visual Basic, and XML. More information on these mappings can be found in *e.g.* [17]. The ORM basics can be summarised as follows: an n-ary predicate (relation) R, with $n \geq 1$, is composed of $r_1, ..., r_n$ roles where each role has one object type, denoted with $C_1, ..., C_n$, associated with it. Roles and predicates are globally unique (although the 'surface labeling' for the modeler may suggest otherwise).

Halpin's first order logic formalization [16] was taken as basis for the mapping into $\mathcal{DLR}_{ifd}$, because it was the first formal characterisation of ORM and is relatively comprehensive in its treatment of constructors; other formalizations of ORM [13, 20, 21] do not differ significantly from Halpin's version. [20, 21] make clearer distinctions between roles and predicates and the relation between them and the naming and labeling of ORM elements, but they cover fewer constraints.

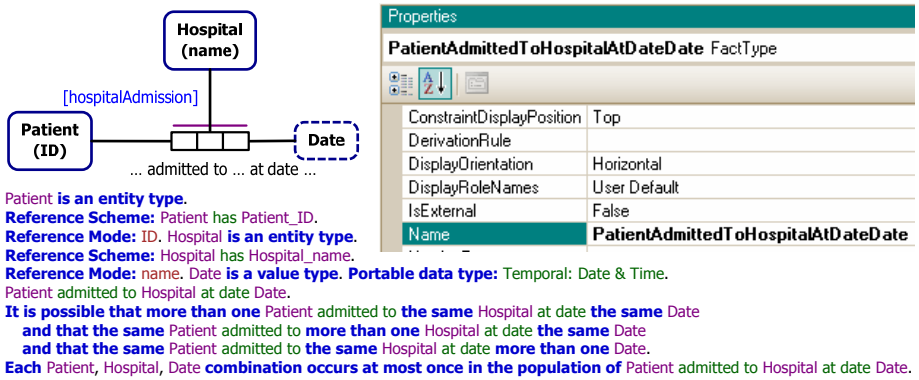

Fig. 1. Top left: an ORM2 model with two object types, a value type, a ternary relation, label for the reading, and name of the first role in "[ ]"; top-right: properties box of the fact type, displaying the name of the relation; bottom-half: verbalization of the fact type, its object and value types, and uniqueness constraint (line above the rectangle).

1.2 DLs for conceptual modelling languages

DL languages are decidable fragments of first order logic and are used for logic-based knowledge representation. Basic ingredients of all DL languages are *concepts* and *roles* (an n-ary predicate where $n \geq 2$). In addition, a DL has several constructors, giving greater or lesser expressivity and efficiency of automated reasoning over a logical theory. The Terminological Box (TBox) contains axioms at the concept-level and the ABox contains assertions about instances. A TBox corresponds to a formal conceptual data model or can be used to represent an ontology. More information and its usage can be found in [2].

For conceptual modelling, we introduce $\mathcal{DLR}$ first [8], and subsequently the "*ifd*" extension [4, 9]. Take atomic relations ($\mathbf{P}$) and atomic concepts A as the basic elements of $\mathcal{DLR}$. We then can construct arbitrary relations with arity ≥ 2 and arbitrary concepts according to the following syntax:

$$\mathbf{R} \longrightarrow \top_n \mid \mathbf{P} \mid (\$i/n : C) \mid \neg\mathbf{R} \mid \mathbf{R}_1 \sqcap \mathbf{R}_2$$
$$C \longrightarrow \top_1 \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists[\$i]\mathbf{R} \mid \leq k[\$i]\mathbf{R}$$

i denotes a component of a relation; if components are not named, then integer numbers between 1 and n_{max} are used, where n is the arity of the relation. k is a nonnegative integer for multiplicity (cardinality). Only relations of the same arity can be combined to form expressions of type $\mathbf{R}_1 \sqcap \mathbf{R}_2$, and $i \leq n$, *i.e.*, the concepts and relations must be well-typed. The semantics of $\mathcal{DLR}$ is specified through the usual notion of interpretation, where $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, and the interpretation function $\cdot^{\mathcal{I}}$ assigns to each concept C a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and to each n-ary $\mathbf{R}$ a subset $\mathbf{R}^{\mathcal{I}}$ of $(\Delta^{\mathcal{I}})^n$, s.t. the conditions are satisfied following Table 1. $\top_1$ denotes the interpretation domain, $\top_n$ for $n \geq 1$ denotes a subset of the n-cartesian product of the domain, which covers all introduced n-ary relations; hence "$\neg$" on relations means difference rather than the complement. The $(\$i/n : C)$ denotes all tuples in $\top_n$ that have an instance of C as their

$\top_n^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}})^n$	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\mathbf{P}^{\mathcal{I}} \subseteq \top_n^{\mathcal{I}}$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$(\neg \mathbf{R})^{\mathcal{I}} = \top_n^{\mathcal{I}} \setminus \mathbf{R}^{\mathcal{I}}$	$(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
$(\mathbf{R}_1 \sqcap \mathbf{R}_2)^{\mathcal{I}} = \mathbf{R}_1^{\mathcal{I}} \cap \mathbf{R}_2^{\mathcal{I}}$	$(\$i/n : C)^{\mathcal{I}} = \{(d_1, ..., d_n) \in \top_n^{\mathcal{I}} \mid d_i \in C^{\mathcal{I}}\}$
$\top_1^{\mathcal{I}} = \Delta^{\mathcal{I}}$	$(\exists[\$i]\mathbf{R})^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists(d_1, ..., d_n) \in \mathbf{R}^{\mathcal{I}}.d_i = d\}$
	$(\leq k[\$i]\mathbf{R})^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \|\{(d_1, ..., d_n) \in \mathbf{R}_1^{\mathcal{I}} \mid d_i = d\}\| \leq k\}$

Table 1. Semantic rules for $\mathcal{DLR}_{\mathit{ifd}}$.

i-th component. $\mathcal{DLR}$ is a proper generalization of $\mathcal{ALCQI}$, where the usual DL constructs can be re-expressed in $\mathcal{DLR}$ as: $\exists P.C$ as $\exists[\$1](P \sqcap (\$2/2 : C))$, $\exists P^{-}.C$ as $\exists[\$2](P \sqcap (\$1/2 : C))$ and so forth (see [8] for details). The following abbreviations can be used: $C_1 \sqcup C_2$ for $\neg(\neg C_1 \sqcap \neg C_2)$, $C_1 \Rightarrow C_2$ for $\neg C_1 \sqcup C_2$, $(\geq k[i]R)$ for $\neg(\leq k-1[i]R)$, $\exists[i]R$ for $(\geq 1[i]R)$, $\forall[i]R$ for $\neg\exists[i]\neg R$, $R_1 \sqcup R_2$ for $\neg(\neg R_1 \sqcap \neg R_2)$, and $(i : C)$ for $(i/n : C)$ when n is clear from the context.

$\mathcal{DLR}_{\mathit{ifd}}$ supports identification assertions on a concept C, which has the form $(\mathbf{id}\ C[i_1]R_1, ..., [i_h]R_h)$, where each R_j is a relation and each i_j denotes one component of R_j. Then, if a is an instance of C that is the i_j-th component of a tuple t_j of R_j, for $j \in \{1, ..., h\}$, and b is an instance of C that is the i_j-th component of a tuple s_j of R_j, for $j \in \{1, ..., h\}$, and for each j, t_j agrees with s_j in all components different from i_j, then a and b are the same object. $\mathcal{DLR}_{\mathit{ifd}}$ supports functional dependency assertions on a relation R for operations, which has the form $(\mathbf{fd}\ R\ i_1, ..., i_h \rightarrow j)$, where $h \geq 2$, and $i_1, ..., i_h, j$ denote components of R.

Other relevant DL languages There are three other $\mathcal{DLR}$ flavours. $\mathcal{DLR}_{\mu}$ supports fixpoint constructs for recursive structures over single-inheritance trees of a role [10] and thereby can represent acyclicity, transitivity, asymmetry, and (ir)reflexivity. $\mathcal{DLR}_{\mathit{reg}}$ adds support for regular expressions over roles (including the role composition operator and reflexive transitive closure) [11], and $\mathcal{DLR}_{\mathcal{US}}$ for temporal EER [1]. It has not been investigated if combining $\mathcal{DLR}_{\mathit{ifd}}$, $\mathcal{DLR}_{\mathit{reg}}$, and $\mathcal{DLR}_{\mu}$ remains within EXPTIME or leads to undecidability. In the other direction toward expressive DL-based ontology languages, OWL and draft OWL 1.1 [26] are based on $\mathcal{SHOIN}$ (for OWL-DL), $\mathcal{SHIF}$ (OWL-Lite), and $\mathcal{SROIQ}$, respectively. $\mathcal{SROIQ}$ supports local (ir)reflexivity, (a)symmetry, and transitive roles [22], but does not have constructors for acyclic roles, datatypes, **id**, and has no 'access' to elements of a DL-role.

2 Mapping issues

We now proceed to the mapping, which considers all components and constraints of ORM2, except deontic constraints (compared to ORM in [16], ORM2 also supports exclusive total covering of subtypes, role values, and deontic constraints). As basis, we used the ORM formalisation in first order logic by [16]. Graphical notation of ORM constrains and more explanation is deferred to [23] due to

space limitations; *e.g.*, the ternary relation in Fig.1 is *reified* in $\mathcal{DLR}_{ifd}$ as: PatientAdmittedToHospitalAtDateDate $\sqsubseteq \exists[1]r_1 \sqcap (\leq 1[1]r_1) \sqcap \forall[1](r_1 \Rightarrow (2$:Patient $))\sqcap \exists[1]r_2 \sqcap (\leq 1[1]r_2) \sqcap \forall[1](r_2 \Rightarrow (2:$ Hospital $))\sqcap \exists[1]r_3 \sqcap (\leq 1[1]r_3) \sqcap \forall[1](r_3 \Rightarrow (2:$ Date $))$ and the identification of Hospital either as (**id** Hospital [1]Hospital-HasHospital_name) or through a 1:1 relation (abbreviated as HHH_n) HHH_n $\sqsubseteq$ (1: Hospital) $\sqcap$ (2: Hospital_name), Hospital $\sqsubseteq (\leq 1$ [1]HHH_n), and Hospital_name $\sqsubseteq (\leq$ 1 [2]HHH_n) with mandatory Hospital $\sqsubseteq \exists$ [1] HHH_n. The interesting problematic constraints are addressed in this section. The main problems concern ORM ring constraints, which are DL-role properties (Fig.2), and constraints with patterns of the type "constraint x over k ORM-roles" over an n-ary relation where $k < n$.

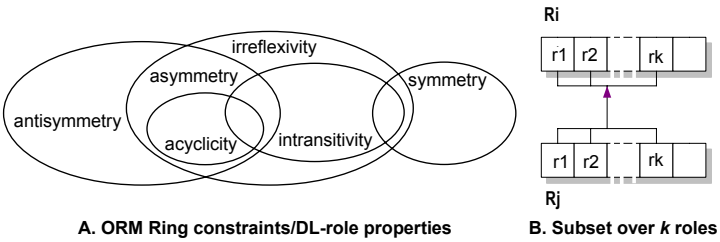

Fig. 2. A: ring constraints (after [17]); B: example of constraint over k ORM-roles.

Intransitivity over an ORM ring constraint is, obviously, supported in $\mathcal{DLR}_{ifd}$, but not transitivity, for which we need either $\mathcal{DLR}_{\mu}$ or $\mathcal{DLR}_{reg}$. Antisymmetry in ORM is *reflexive antisymmetry* $(\forall x, y(R(x,y) \wedge R(y,x) \rightarrow x = y))$, which no DL language supports. (Observe from Fig.2 that $\mathcal{SROIQ}$'s irreflexive antisymmetry is asymmetry.) The irreflexive ring constraint on a binary relation $(\forall x \neg(R(x,x)))$ is an open issue for $\mathcal{DLR}_{ifd}$, but already possible with $\mathcal{DLR}_{\mu}$ thanks to least/greatest fixpoint construct and in $\mathcal{SROIQ}$ with `Self`. The symmetric $(\forall x, y(R(x,y) \rightarrow R(y,x)))$ and asymmetric $(\forall x, y(R(x,y) \rightarrow \neg R(y,x)))$ ring constraints are not supported either, but both are supported in $\mathcal{SROIQ}$ and the latter is supported in $\mathcal{DLR}_{\mu}$ through the stronger notion of well-foundedness. The last 'basic' ring constraint, acyclicity ("R is acyclic iff $\forall x \neg(x$ has path to $x)$" in [17]), probably can be added to $\mathcal{DLR}_{ifd}$ with the repeat PDL (transitive closure of roles, R^+, *i.e.*, $\bigcup_{n\geq 1}(R^{\mathcal{I}})^n$) using the least fixpoint construct $\mu X.C$ (*i.e.*, $\exists R^*.C = \mu X(C \sqcup \exists R.X)$ [8, 10]). ORM also permits *combinations* of ring constraints: intersecting acyclicity and intransitivity, antisymmetry with intransitivity, intransitive symmetry, and irreflexive symmetry.

The second main problem concerns constraints over k roles in an n-ary relation, which are: *Subset over k roles* in two n-ary relations (depicted in Fig.2-`B`, `A` below with abbreviation that underlined variable stands for a sequence $x_1, ..., x_n$ in an n-ary relation [23]), $k < n$, where the corresponding roles must match in domain, *Set-equality over k roles* (`B`), *Role exclusion over k roles* (`C`) in two n-ary relations R_i and R_j, and *Multi-role frequency* spanning i roles of an n-ary

relation, $i > 2$, and $i \leq n$ (TFC5 in [16]), with formalisms as follows [16].
`A.` $\forall x_1, ...x_n(\exists\ \underline{y}\ (R_j(\underline{y})\ \wedge x_1 = y_{j_1} \wedge ... \wedge x_n = y_{j_n}) \rightarrow \exists\ \underline{z}\ (R_i(\underline{z}) \wedge x_1 = z_{i_1} \wedge ... \wedge x_n = z_{i_n}))$ `// Subset over` k `roles`
`B.` $\forall x_1, ...x_n(\exists\ \underline{y}\ (R_j(\underline{y})\ \wedge x_1 = y_{j_1} \wedge ... \wedge x_n = y_{j_n}) \equiv \exists\ \underline{z}\ (R_i(\underline{z}) \wedge x_1 = z_{i_1} \wedge ... \wedge x_n = z_{i_n}))$ `// Set-equality over` k `roles`
`C.` $\forall x_1, ..., x_n \neg(\exists \underline{y}(R_i(\underline{y}\ \wedge x_1 = y_{i_1} \wedge ... \wedge x_n = y_{i_n}) \wedge \exists\ \underline{z}\ (\ R_j(\underline{z}) \wedge x_1 = z_{j_1} \wedge ... \wedge x_n = z_{j_n}))$ `// Role exclusion over` k `roles`
The problem is that these constraints lead to undecidability if those k roles do not exactly make up the primary key (spanning uniqueness), as in a relational table (`A`)-(`C`) correspond to arbitrary projections. If one does not consider an additional key constraint, a *partial* mapping of (`A`)-(`C`) can be made on a ORM role-by-role basis (DL-role element by DL-role element). This can be reduced to a minor issue for *Multi-role frequency* (`D`) spanning roles r_i, r_j in an n-ary relation, $n \geq 2$, and $1 \leq a \leq b$ and subsequently assesses it in combination with permissible uniqueness constraints.
`D.` $\forall x, y(\exists z_1 R(x, y, z_1) \rightarrow \exists z_2, ..., z_a(z_1 \neq z_2 \wedge ... \wedge z_{a-1} \neq z_a \wedge R(x, y, z_2) \wedge ... \wedge R(x, y, z_a))) \wedge \forall x, y, z_1, ..., z_{b+1}(R(x, y, z_1) \wedge ... \wedge R(x, y, z_{b+1}) \rightarrow z_1 = z_2 \vee z_1 = z_3 \vee ... \vee z_b = z_{b+1})$ `// Multi-role frequency`
Given that an elementary fact type must have uniqueness over n-1 roles, then either 1) r_i or r_j is part of a single role uniqueness constraint but not both, 2) r_i or r_j is part of a multi-role uniqueness constraint but not both, 3) multi-role uniqueness includes r_i, r_j, and ≥ 1 other role in the relation, or 4) the relation is not an elementary fact type and ought to be remodelled to be elementary. Option 1 implies that either i) $a = 1$ or ii) $b = 1$, and then the constraint can be reduced to 1:n and m:n uniqueness, respectively; options 2-4, however, reduce to the same problem of undecidability as with (C) (see [23]).

Last, a minor mapping issue is the *Role value constraint*, which is new in ORM2: object type C_i only participates in role r_i if an instance has any of the values $\{v_i, ...v_k\}$, which is a subset of the set of values C_i can have. With a binary relation, then $\forall x, y(x \in \{v_i, ..., v_k\} \rightarrow (R(x, y) \rightarrow C_i(x) \wedge C_j(y)))$ holds. A 'candidate approach' is to try to use $\mathcal{DLR}_{\mathit{ifd}}$ through breaking down the constraint by creating a new subtype C_i' for the set of values to which the role is constrained, where the value can be any of $\{v_i, ...v_k\}$, and let C_i' play role r_i, s.t. $C_i' \sqsubseteq C_i$ and $C_i' \sqsubseteq \forall[r_i]R$ and then use named value types for the value constraints on C_i'. Alternatively, remodelling with role values might be an option, but, at present, this is supported only in *DL-Lite*$_{\mathcal{A}}$ [6].

3 Discussion and Conclusions

As has become clear from the mapping, the ORM ring constraints/DL-role properties are most problematic for $\mathcal{DLR}_{\mathit{ifd}}$, but most of them can be met by $\mathcal{DLR}_{\mu}$ or $\mathcal{SROIQ}$. On the other hand, $\mathcal{DLR}_{\mu}$ and $\mathcal{SROIQ}$ do not have a constructor for non-unary primary keys, and $\mathcal{SROIQ}$ neither supports n-ary relations where $n > 2$ nor provides a means to 'access' an ORM-role/DL-role element. For these reasons, $\mathcal{DLR}_{\mathit{ifd}}$ was chosen. Another advantage of having taken $\mathcal{DLR}_{\mathit{ifd}}$

is that the syntax of UML class diagrams have been mapped into it [4], thereby augmenting the current informal mapping in [17] and moving closer to interoperability between ORM and UML through a *formal* correspondence between the two conceptual modelling languages with $\mathcal{DLR}$ as unifying paradigm.

Looking toward implementations, EER and UML were mapped to $\mathcal{DLR}$ and $\mathcal{DLR}_{ifd}$ earlier and implemented in the iCOM tool [15], where one can use tools like Racer, Pellet, and FaCT. Hence, it uses a $\mathcal{SHIQ}$-based reasoner through an additional transformation step; note that there are differences between theoretically computationally feasible and actually implemented reasoning services. Nevertheless, reasoning over less complex ORM-models is already a considerable advantage over no automated reasoning services at all. Moreover, if iCOM adds a module for ORM support, the modeller would be able to describe already supported inter-model assertions between EER, UML, and, now, ORM models and reason over any combination. Further down the line, a software developer will benefit from a better, consistent and known to be satisfiable, conceptual model and with *e.g.* NORMA would be able take advantage of the already implemented features of automated generation of relational databases and of software code.

Concluding, most –and the most often used– ORM2 elements and constraints can be mapped into $\mathcal{DLR}_{ifd}$. This already could be used for a wide range of ORM-models that do not use its full expressive capabilities; *e.g.*, to carry out model checking, compute derived relations, and classification. Conversely, when the present mapping is implemented, DLs will have a sophisticated user interface enabling domain experts to take part in representing their Universe of Discourse. Several approaches are possible to narrow the gap between ORM2 and DL languages, where a "$\mathcal{DLR}_{\mu ifd}$" or $\mathcal{SROIQ}$ with n-ary relations seem close by. But to take advantage of narrowing the gap, tools for automated reasoning services will have to expand their features list as well. Alternatively, if this leads to undecidability or intractability, one could investigate modularization where a large conceptual model can be split-up into sections (ideally, hidden from the modeller) and perform the reasoning on the separate subsections. We are currently working on formal proofs of the 'mismatches' between ORM and the $\mathcal{DLR}$s.

References

1. Artale, A., Franconi, E., Wolter, F., Zakharyaschev, M. A temporal description logic for reasoning about conceptual schemas and queries. In: *Proc. of JELIA '02*, S. Flesca, S. Greco, N. Leone, G. Ianni (Eds.), Springer, 2002, LNAI 2424, 98-110.
2. Baader, F. Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds). *Description Logics Handbook*, Cambridge University Press, 2003.
3. Balsters, H., Carver, A., Halpin, T., Morgan, T. Modeling dynamic rules in ORM. *2nd International Workshop on Object-Role Modelling (ORM 2006)*. LNCS 4278. Berlin: Springer-Verlag, 2006, 1201-1210.
4. Berardi, D., Calvanese, D., De Giacomo, G. Reasoning on UML class diagrams. *Artificial Intelligence*, 2005, 168(1-2):70-118.
5. Bollen, P. Using fact-orientation for instructional design. *2nd International Workshop on Object-Role Modelling (ORM 2006)*. LNCS 4278. Berlin: Springer-Verlag, 2006, 1231-1241.

6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati. Linking data to ontologies: The description logic DL-Lite A. In *Proc. of the 2nd Workshop on OWL: Experiences and Directions (OWLED 2006)*, 2006.
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R. DL-Lite: Tractable description logics for ontologies. In: *Proc. of AAAI 2005*, 602-607.
8. Calvanese, D., De Giacomo, G. Expressive description logics. In: *The Description Logic Handbook: Theory, Implementation and Applications*, Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (Eds). Cambridge University Press, 2003. 178-218.
9. Calvanese, D., De Giacomo, G., Lenzerini, M. Identification constraints and functional dependencies in Description Logics. In *Proc. of IJCAI 2001*, 2001, 155-160.
10. Calvanese, D., De Giacomo, G., Lenzerini, M. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In: *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, 84-89, 1999.
11. Calvanese, C., De Giacomo, G., Lenzerini, M. On the decidability of query containment under constraints. In: *Proc. of PODS'98*, 149-158, 1998.
12. Calvanese, D., Lenzerini, M., Nardi, D. (1998) Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*. Kluwer, Amsterdam.
13. Campbell, L.J., Halpin, T.A. and Proper, H.A. Conceptual Schemas with Abstractions: Making flat conceptual schemas more comprehensible. *Data & Knowledge Engineering*, 1996, 20(1): 39-85.
14. Evans, K. Requirements engineering with ORM. *Intl. Workshop on Object-Role Modeling (ORM'05)*. LNCS 3762. Berlin: Springer-Verlag, 2005. 646-655.
15. Franconi, F., Ng, G. The ICOM Tool for Intelligent Conceptual Modelling. *7th Workshop on Knowledge Representation meets Databases (KRDB'00)*, Berlin, 2000.
16. Halpin, T.A. *A logical analysis of information systems: static aspects of the data-oriented perspective*. PhD Thesis, University of Queensland, Australia. 1989.
17. Halpin, T. *Information Modeling and Relational Databases*. San Francisco: Morgan Kaufmann Publishers, 2001.
18. Halpin, T. Objectification of relationships. *10th Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD'05)*. Porto, Portugal, 2005.
19. Halpin, T. ORM2. *International Workshop on Object-Role Modeling (ORM'05)*. LNCS 3762. Berlin: Springer-Verlag, 2005. 676-687.
20. Hofstede, A.H.M. ter, Proper, H.A., Weide, Th.P. van der. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 1993, 18(7):489-523.
21. Hofstede, A.H.M. ter, Proper, H.A.. How to Formalize It? Formalization Principles for Information Systems Development Methods. *Information and Software Technology*, 1998, (40(10): 519-540.
22. Horrocks, I., Kutz, O., Sattler, U. The Even More Irresistible $\mathcal{SROIQ}$. In: *Proceedings of KR-2006*, Lake District, UK, 2006.
23. Keet, C.M. *Mapping the Object-Role Modeling language ORM2 into Description Logic language* $\mathcal{DLR}_{ifd}$. Technical Report KRDB07-2, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy. 15 February 2007. arXiv:cs.LO/0702089v1.
24. Pepels, B., Plasmeijer, R. Generating applications from Object Role Models. *International Workshop on Object-Role Modeling (ORM'05)*. LNCS 3762. Berlin: Springer-Verlag, 2005, 656-665.
25. NORMA. http://sourceforge.net/projects/orm/.
26. OWL 1.1 http://owl1_1.cs.manchester.ac.uk/owl_specification.html.
27. OWL. http://www.w3.org/TR/2004/REC-owl-semantics-20040210/syntax.html.