Modeling ORM Schemas in Description Logics

Thi Dieu Thu NGUYEN* and Nhan LE THANH and TA Le PHAM

I3S laboratory (CNRS - UNSA), Les Algorithmes - Euclide B Building, 2000 Route des Lucioles – B.P. 121, F-06903 Sophia Antipolis Cedex, France.

Abstract. In recent years, there has been a growing interest in integration of semantics into the Semantic Web environment, whose goal is to access, relate and combine information from multiple sources. With regard to this tendency, our work studies a mechanism to model ORM schemas in the Description Logic language $\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$, the underpinning of a Web ontology language. This mechanism meets the key feature required by ORM schemas (i.e. identification and functional dependency constraints). It can be applied to integrate information not only from systems described in ORM schemas but also from relational databases into the Semantic Web environment.

Key words: Information integration, object role modeling (ORM), description logics (DLs), web ontology language, semantic web.

1 Introduction

In recent years, the problem of interoperability and semantic integration of heterogeneous information sources in the Semantic Web environment has triggered various research [2, 11, 15]. Besides the universal environment of the WWW (World Wide Web), this is thanks to the expressivity of Web ontology languages that deals with the problem of heterogeneity of sources and to their well-formalized annotations which can be used in automated reasoning [9]. These features are provided by underlying description logic (DL) languages, a family of knowledge representation formalisms based on First-Order Logic (FOL) [17], e.g. $\mathcal{SHOIN}(\mathbf{D})$ [7, 8].

One of the most popular information modeling approaches nowadays is ORM (Object Role Modeling) [4, 5]. It facilitates modeling, transforming, and querying information using *facts* and *constraints*, which may be verbalized in a close-to-natural language. This increases the ease of use, specially toward non-technical users. Unlike Entity-Relationship (ER) modeling [19] and Unified Modeling Language (UML) [16], ORM is attribute-free. The latter avoids the problems caused by instability in ER and UML as clearly shown in [3].

* Tel: +33 (0)492942743; Fax: +33 (0)492942898; Email: tnguyen@i3s.unice.fr

Moreover, for information modeling, ORM graphical notations are far more expressive than those of ER and UML. It provides procedures for mapping to other database schemas (e.g. ER, UML). ORM has been fully formalized in FOL [4], so that its semantics is very close to that of DLs.

Hence, with the aim of integrating information systems into the Web Semantics, we introduce a formalization of ORM schemas in terms of a DL. In particular, we show how ORM schema semantics can be captured in Web Semantics thanks to the expressivity of the DL $\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$ [14, 12], which in its turn is the underpinning of a Web ontology language [13].

Several formalizations of ORM schemas have been proposed in the literature [10, 18]. They have been proved very useful with respect to establishing a common understanding of the formal meaning of ORM schemas. However, to the best of our knowledge, none of them has the explicit goal of building a framework to integrate information into the Semantic Web environment.

The rest of the paper is organized as follows. Section 2 briefly introduces the $\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$ language. In section 3, we show how ORM schemas can be formalized in this DL. Section 4 will conclude the paper with some future perspectives.

2 Description Logic $\mathcal{SHOIN}\mathcal{K}$(D)

2.1 Concrete Datatypes

Concrete datatypes are used to represent literal values such as numbers or strings. They compose a *concrete domain* $\mathbf{D}$, as introduced for $\mathcal{SHOQ}(\mathbf{D})$ [7]. Each datatype $d \in \mathbf{D}$ is associated with a set $d^{\mathbf{D}} \subseteq \Delta_{\mathbf{D}}$, where $\Delta_{\mathbf{D}}$ is the domain of interpretation of all datatypes. For example, a datatype $\geq_{21}$ in $\mathbf{D}$ defines a set ${\geq_{21}}^{\mathbf{D}}$ of *integer* values greater than or equal to 21.

2.2 $\mathcal{SHOIN}\mathcal{K}$(D) Syntax and Semantics

$\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$, presented in [14, 12], is an extension of $\mathcal{SHOIN}(\mathbf{D})$ with a decidable inference by adding identification constraints (ICs), which are one-to-one relations between an identifying and the identified part.

As a DL language, the basic syntactic building blocks of $\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$ are atomic *concepts*, atomic *roles*, and individuals (constants). Concepts are interpreted as sets of individuals (subsets of the interpretation domain) and roles as sets of pairs of individuals. Expressions are then built from these basics by using several kinds of constructors. For example, the *conjunction* $C \sqcap D$ denotes the set of individuals obtained by intersecting the sets of individuals of C and D. In FOL, this expression can be described as $C(x) \wedge D(x)$, where the variable ranges over all individuals in the interpretation domain and $C(x) \wedge D(x)$ is true for those belonging to the concept C and D.

Syntax and Semantics of $\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$ can be seen in table 1, where C, D are concept descriptions; o is nominal, i.e. singleton concept; $R, R_1, ..., R_n$, S are roles; $\mathbf{R}_+$ is the set of transitive roles; $\sharp$ denotes cardinality; $\mathcal{I}$ is the interpretation function; $\Delta^{\mathcal{I}}$ is the interpretation domain disjoint from $\Delta_{\mathbf{D}}$, the concrete interpretation domain. Further details of $\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$ can be found in [14, 12].

Table 1. $\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$ Syntax and Semantics

Construct name	**Syntax**	**Semantics**
atomic abstract role	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
concrete role	U	$U^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}_{\mathbf{D}}$
transitive role	$R \in \mathbf{R}_+$	$R^{\mathcal{I}} = R^{\mathcal{I}+}$
inverse abstract role	R^-	$\{\langle x, y\rangle \mid \langle y, x\rangle \in R^{\mathcal{I}}\}$
role hierarchy	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
datatype	d	$d_{\mathbf{D}} \subseteq \Delta^{\mathcal{I}}_{\mathbf{D}}$
concrete value	v	$v^{\mathcal{I}} = v^{\mathbf{D}}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
exists restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y. \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
value restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y. \langle x, y\rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
concrete exists restriction	$\exists R.d$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y. \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in d^{\mathbf{D}}\}$
concrete value restriction	$\forall R.d$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y. \langle x, y\rangle \in R^{\mathcal{I}} \rightarrow y \in d^{\mathbf{D}}\}$
atleast restriction	$\geq nR$	$\{x \in \Delta^{\mathcal{I}} \mid \sharp\{y. \langle x, y\rangle \in R^{\mathcal{I}}\} \geq n\}$
atmost restriction	$\leq nR$	$\{x \in \Delta^{\mathcal{I}} \mid \sharp\{y. \langle x, y\rangle \in R^{\mathcal{I}}\} \leq n\}$
nominal	o	$\sharp\{o^{\mathcal{I}}\} = 1$
subsumption	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
identification constraint	$(R_1, ..., R_n$ **Idfor** $C)$	$\mathcal{I} \models (R_1, ..., R_n$ **Idfor** $C)$ iff $\forall s, s' \in C^{\mathcal{I}}$ and $\langle s, t_i\rangle, \langle s', t'_i\rangle \in R_i^{\mathcal{I}}\ \forall 1 \leq i \leq n$, $t_i = t'_i\ \forall 1 \leq i \leq n$ then $s = s'$

As shown in table 1, $\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$ concept descriptions can be specified as logical combinations (conjunction, disjunction, or negation) of others, as nominals or as restrictions on a particular role so that all the values filling the role have to belong to a certain concept (or datatype); or at least one value must come from a certain concept (or datatype); or there is at least or at most a certain number of distinct values. Concept descriptions and roles can also be organized in a subsumption (subclass) hierarchy. $\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$ can also state whether a role is transitive, or the inverse of another.

3 Modeling ORM in $\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$

Example 1 is an illustration of modeling ORM schemas in $\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$. We use the ORM formalization and syntax as found in [4, 5, 6], but, limiting it to

binary predicates[2]. In this work, we consider the critical constraints that are indispensable in designing ORM schemas and we do not support nested-fact types and derivation constraints (the latter are not part of the ORM graphical notation).

Example 1. Fig. 1 describes in ORM graphical notation [6] the information of the students who attended an International meeting of the best students in the world. Each country, coded by CountryId, had only one student per subject (which is coded by SubjectId) who was elected to represent his/her country in this discipline. For social events, each student could attend many groups, each of which has only one topic. Different groups can share the same topic, but the student can only attend groups having different topics. These descriptions can be expressed in $\mathcal{SHOINK}(\mathbf{D})$ as shown in Table 2.

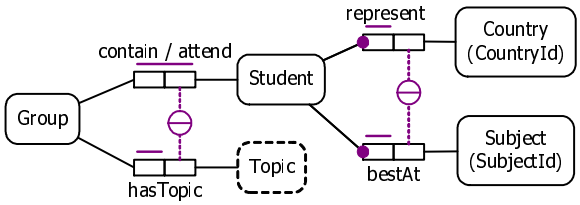

Fig. 1. ORM schema example

The principle components of ORM are object types, which include entities (e.g. Student, Country) and values (e.g. Topic, CountryId), and roles (e.g. contain, represent). Object types are connected by the roles they play, composing the fact types. Two roles in a fact type make up its binary predicate. Constraints are applied to roles to create the population of a database and restrict the data permitted in it. We here focus on the representation of the mandatory (•), internal uniqueness (—) and external uniqueness (⊖) constraints. We briefly explain the formalization shown in Table 2, where $C_1, C_2, ..., C_n$ are entity names; $R_1, R_2, ..., R_n$ are role names; $(CR_i/R_{inv(i)}C_i)$ are fact types, $R_{inv(i)}$ is the inverse of R_i, C is an entity name and $1 \leq i \leq n$; C(Id) is the reference scheme of C; R_{id} is a role name used to translate a reference scheme; C_{fd} and R_{fd} are respectively an entity and a role name used to translate a functional dependency.

The basic process is to go through a given schema, just as an automated system would do, looking for the following descriptions in the schema:

1. Object types (entities and values)
2. Reference schemes
3. Roles
4. Constraints

[2] It has been proved that n-ary predicates can be transformed to binary ones [20]

a) Identification constraints on one role
b) Identification constraints on many roles
c) Functional dependencies on many roles
d) Other constraints

It is obvious that this mapping is polynomial w.r.t the number of elements in the schema.

3.1 Mapping Fact Types

Since constraints are applied to fact types, first of all, object types and roles must be formalized.

Values are object types that cannot be defined by others. They are either strings or numbers, which exactly correspond to datatypes in $\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$. Entities can be defined by other entities and/or values. When defined by values, they can be seen as atomic concepts in $\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$. Otherwise, constraints must be used to describe them (see section 3.2).

A role can attribute a value to an entity (attributive role, e.g. $\mathsf{hasTopic}$) or connect one entity to another (e.g. $\mathsf{represent}$). The former corresponds to a concrete role while the latter corresponds to an abstract role in $\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$. In any case, the instances assisting a given fact compose a set subsumed by their entity. So that we can model the context of the role they play as an axiom where the set generated by *atleast* restriction by 1 on the role is subsumed by the entity playing the role (e.g. $\geq 1\mathsf{hasTopic} \sqsubseteq \mathsf{Group}$). Besides, whenever the entity plays its role, the object it plays with is defined by the role. So we can formalize this expression by an axiom specifying that the *value* restriction on the role subsumes the universe (e.g. $\top \sqsubseteq \forall \mathsf{hasTopic}.\mathsf{Topic}$). In binary predicates, one role needs to be the inverse of the other. Hence, for a given fact type, two predefined non-attributive roles are mapped as the inverse of each other (e.g. $\mathsf{attend} \equiv \mathsf{contain}^{-}$).

3.2 Mapping Constraints

A mandatory constraint (MC) on a role specifies that all instances of an object type must play the role. For example, it is applied on the role $\mathsf{represent}$ in Example 1 to show that every student must represent some country. This constraint is equally expressed by the existence restriction in $\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$.

Uniqueness constraints (UCs) are used to express that each instance of an object type plays a set of roles at most once. These constraints make think of *atmost* restrictions in $\mathcal{SHOIN}\mathcal{K}(\mathbf{D})$. However, *atmost* restrictions are applied to a single role only. UCs in this case (e.g. on $\mathsf{represent}$, bestAt) can thus be described by *atmost* restrictions by 1.

In case UCs are applied to a set of two roles of the same predicate (e.g. $\mathsf{contains}/\mathsf{attend}$), the two objects are in a n-n relationship (in Example 1, the objects are Group and $\mathsf{Student}$). Implicitly, a binary relation is n-n without condition. So that there is no need to make a translation for this case.

Table 2. Modeling ORM schemas in $\mathcal{SHOINK}(\mathbf{D})$

ORM	$\mathcal{SHOINK}(\mathbf{D})$	**Example**
Value C_i	Datatype C_i	Topic
Entity C_i	Concept C_i	Group
Role R_i	$\geq 1 R_i \sqsubseteq C$	$\geq$ 1contain $\sqsubseteq$ Group
	$\top \sqsubseteq \forall R_i.C_i$	$\top \sqsubseteq \forall$contain.Student
Inverse role $R_{inv(i)}$	$R_{inv(i)} \equiv {R_i}^-$	attend $\equiv$ contain$^-$
MC on R_i	$C \sqsubseteq \exists R_i.C_i$	Student $\sqsubseteq \ \exists$ bestAt.Subject
IUC on R_i	$\mathsf{C} \sqsubseteq \leq 1\ \mathsf{R}_i$	Group $\sqsubseteq \ \leq 1$ hasTopic
Reference scheme	$\geq 1\ \mathsf{R}_{id} \sqsubseteq \mathsf{C}$	$\geq$ 1idForCountry $\sqsubseteq$ Country
C(Id)	$\top \sqsubseteq \forall\ \mathsf{R}_{id}$.Id	$\top \sqsubseteq \forall$idForCountry.CountryId
(simple IC)	(R_{id} **Idfor** C)	(idForCountry **Idfor** Country)
EUC on ($\mathsf{R_1}^-$,..., $\mathsf{R_n}^-$)	(R_1,..., R_n **Idfor** C)	(represent, bestAt **Idfor** Student)
(IC for C)		
EUC on	$\geq 1 R_{fd} \sqsubseteq C_{fd}$	$\geq$ 1specGroup $\sqsubseteq$ StudentTopic
($\mathsf{R_1}^-$, $\mathsf{R_2}^-$, ..., $\mathsf{R_n}^-$)	$\top \sqsubseteq \forall R_{fd}.\mathsf{C}$	$\top \sqsubseteq \forall$specGroup.Group
(FD for C)	$C_{fd} \sqsubseteq \leq 1 R_{fd}$	StudentTopic $\sqsubseteq \ \leq 1$ specGroup
	(R_1,...,R_n **Idfor** C_{fd})	(contain, hasTopic **Idfor** StudentTopic)

Fig. 2. Description of the reference scheme Country(CountryId)

In Fig. 1, we also see entities as so called *reference schemes*, e.g. Country(CountryId). The latter specifies that CountryId identifies uniquely Country. Essentially, reference schemes can be explained by adding an attributive role, imposing an MC and two UCs on both the two roles in the predicate (see Fig. 2). However, the inverse of an attributive role cannot be represented in $\mathcal{SHOINK}(\mathbf{D})$ (see Table 1). Nevertheless, ICs in $\mathcal{SHOINK}(\mathbf{D})$ allow us to represent this ORM formalization, e.g. (idForCountry **IdFor** Country).

This combination of constraints can also be described explicitly in ORM schemas as ICs. In such cases, the same mapping mechanism is applied.

The UCs discussed above are applied to roles in a single predicate. They are called *internal uniqueness constraints* (IUCs). In case applied to the roles in different predicates, these constraints, the so-called *external uniqueness constraints* (EUCs), are used to express unique identification and functional dependency (FD) relationships:

- *Using EUCs to describe unique identifications.* In Example 1, the EUC on the two inverse roles of represent and bestAt, combined with the IUCs and the MCs on the two latter roles, describe that students are uniquely identified by their country and their subject at which they were the best. Hence, the combination of these constraints can be interpreted as an IC in $\mathcal{SHOINK}(\mathbf{D})$, i.e., (represent, bestAt **Idfor** Student).

- *Using EUCs to describe FDs.* Except the case described above, the application of EUCs results in FDs. For example, the EUC on attend and the inverse of hasTopic specifies that Group is functionally dependent on the couple (Student,Topic). Note that in $\mathcal{SHOINK}(\mathbf{D})$, constructors, except for ICs, are applied to one role only. In that way, it is not capable of expressing FDs nor ICs on many roles [1]. However, observe that an FD of an object A on two objects B and C can be described as an FD of A to an object A', which is uniquely identified by the couple (B,C). Hence, we make use of this idea to formalize this kind of EUCs. For example, modeling the given EUC on attend and the inverse of hasTopic, we create an entity StudentTopic which is uniquely identified by the couple (Student,Topic). A new role, specGroup, is created to set the functional dependency of Group on StudentTopic. The two ORM schemas in Fig. 3 are equivalent.

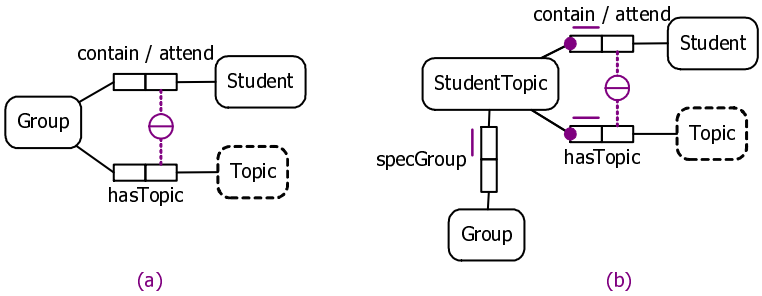

Fig. 3. Equivelent schemas for an FD in Example 1

4 Conclusion and Perspectives

We have proposed a new formalization of ORM schemas in the DL $\mathcal{SHOINK}(\mathbf{D})$, which meets the key feature required by the schemas (i.e., identification and functional dependency constraints). Exploiting such a formalization, one can integrate information into the Semantic Web environment from systems described in ORM schemas as well from relational databases. Furthermore, the DL used here is decidable [14, 13]. So this formalization allows for automated reasoning on ORM schemas as well (e.g., detecting constraint conflicts or implications).

Our future work will include the mapping of other types and constraints in ORM to $\mathcal{SHOINK}(\mathbf{D})$ and the implementation of the formalization. With the perspective to construct a methodology of integration of information sources into the Semantic Web environment, our next phase will be the automated translation of requests from a DL to SQL by using this formalization.

References

1. A Borgida and Grant E Weddell. Adding uniqueness constraints to description logics (preliminary report). In *DOOD '97: Proceedings of the 5th International Conference on Deductive and Object-Oriented Databases*, pages 85–102, London, UK, 1997. Springer-Verlag.
2. Li Ding, P Kolari, Z Ding, and S Avancha. *Using Ontologies in the Semantic Web: A Survey*. Springer, October 2005. UMBC CS Technical Report 05-07.
3. TA Halpin. Business rules and object rolev modeling. Available at: http://www.orm.net/pdf/dppd.pdf.
4. TA Halpin. *A logical analysis of information systems: static aspects of the data-oriented perspective*. PhD thesis, University of Queensland, Australia, 1989.
5. TA Halpin. Object-role modeling (ORM/NIAM). In *Handbook on Architectures of Information Systems, 2nd edition*, pages 81–103. Springer, Heidelberg, 2006.
6. TA Halpin. ORM 2 graphical notation. Technical report, Newmont University, September 2005.
7. I Horrocks and U Sattler. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 199–204. Morgan Kaufmann, Los Altos, 2001.
8. I Horrocks and U Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 448–453, 2005.
9. I Horrocks, Patel P Schneider, and F van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
10. M Jarrar and E Franconi. Formalizing ORM using the DLR description logic. 2007. Submitted.
11. Y Kalfoglou, Bo Hu, D Reynolds, and N Shadbolt. Semantic integration technonolgies survey. Technical report, University of Southampton, 2005.
12. TD Thu Nguyen and N Le-Thanh. La contrainte d'identification dans la logique de description $\mathcal{SHOIN}$(**D**). ISRN I3S/RR- 2006-34-FR, Laboratoire I3S (CNRS - UNSA), France, 2006.
13. TD Thu Nguyen and N Le-Thanh. Extending OWL-DL with identification constraints. ISRN I3S/RR- 2007-03-FR, Laboratoire I3S (CNRS - UNSA), France, 2007.
14. TD Thu Nguyen and N Le-Thanh. Identification constraints in $\mathcal{SHOIN}$(**D**). In *Proc. of the 1th Int. Conf. on Research Challeges in Information Science (RCIS 2007)*, 2007. Accepted to publish.
15. Natalya F Noy. Semantic integration: a survey of ontology-based approaches. *SIGMOD Rec.*, 33(4):65–70, 2004.
16. Keng Siau and TA Halpin, editors. *Unified Modeling Language: Systems Analysis, Design and Development Issues*. Idea Group, 2001.
17. Raymond M Smullyan. *First-Order Logic*. Dover Publications, 1995.
18. P Spyns, SV Acker, M Wynants, M Jarrar, and A Lisovoy. Using a Novel ORM-Based Ontology Modelling Method to Build an Experimental Innovation Router. In *EKAW*, pages 82–98, 2004.
19. B Thalheim. *Entity-Relationship Modeling: Foundations of Database Technology*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.
20. JJVR Wintraecken. *The NIAM Information Analysis Method: theory and practice*. Kluwer Academic Publishers, 1990.