ORM to OWL 2 DL Mapping

Rami Hodrob
Arab American University, Jenin, Palestine
Birzeit University, Palestine
rhodrob@aauj.edu

Mustafa Jarrar
Birzeit University, Palestine
mjarrar@birzeit.edu

Abstract: The goal of this article is to map between Object Role Modeling (ORM) and Ontology Web Language 2 (OWL 2 DL). This mapping allows one to graphically develop his/her ontology using the ORM notation, while the ORM is automatically translated into OWL 2 DL. We map the most commonly used rules of ORM into OWL 2 DL which have the ability of decidability. DogmaModeler is extended to perform automatically this mapping (ORM into OWL 2 DL). Mapping technique is assessed using desirable reasoning methodology which depends on RacerPro2 reasoner .

Keywords: Ontology, Object Role Modeling, Web Ontology Language 2 (OWL 2 DL), SHOIN Description Logic.

1. Introduction and Motivation

Ontology is receiving an increasing interest in many application areas such as data integration, semantic web, knowledge engineering and enhanced information retrieval, etc [5]. This led World Wide Web Consortium (W3C) to recommend Ontology Web Language (Owl) [14]. It is difficult for IT people to build an ontology .At the other hand building ontology is time consuming. One who builds ontology needs good knowledge in formal logic.
Building an ontology using graphical notation tool is easier than other available techniques, even for non-IT specialists such as Object Role Modeling (ORM). ORM is a conceptual modeling language used in ontology engineering [13]. It encompasses a group of constraints that can comprehensively represent an ontology using rich graphical notation [7,8]. On the other hand, OWL 2 DL [16] is relatively a non user friendly language to be used by even IT specialists.
In our research, we map between ORM and OWL 2 DL. In this way (mapping) we exploit the benefits of both ORM and OWL 2. The benefits of ORM are i) it is true conceptual modeling independent of application; ii) it is a very user-friendly methodology; iii) it is more expressive than other techniques such as ER and UML [11,2,6]; iv) it is easy to reason about [10]; v) it is used in ontology standards and for expressing business rules [1]. In the other hand the benefits of OWL 2 are i) it is the recommended ontology web language [17]; ii) it is used to publish and share ontologies on the Web semantically; iii) it is used to construct a structure to share information standards for both human and machine consumption; iv) Automatic reasoning can be done against ontologies represented in OWL 2 to check consistency and coherency of these ontologies. That is a good motivation to combine ORM and OWL 2. This way we can build our ontology in ORM which is very close to natural language and easy to understand and use. In other words, we can build a system that uses ORM as interface for OWL 2.
We extend DogmaModeler [4] tool to implement our mapping (ORM into OWL 2) work. Another goal for the mapping is to extend ORM to represent notations that are not supported in ORM and available in OWL 2 like equivalent classes, data types, transitive closure, intersection and union between relations.
As a related work, the mapping from ORM to SHOIN/OWL description logic has been implemented [11]. SHOIN is chosen to compromise both its ability of expressiveness and decidability. Each rule of ORM which is supported by SHOIN is mapped to SHOIN. Twenty two cases of ORM constructs are mapped. The purpose of this research [11] also is to use ORM as a technique and expressive notation for ontology engineering. Although in this research [11] mapping ORM into SHOIN is achieved, but mapping ORM into OWL is not achieved, where in our research mapping ORM into OWL 2 DL is achieved and is implemented automatically. Some other related work to our ontology modeling considered using UML as front-end to visualize and edit ontologies [2] without semantics as we do in our work, in addition the mentioned related work does not map to OWL 2 DL or even to OWL 1.
The rest of the paper is structured as follows. Section 2 briefly describes ORM background. Section 3 describes OWL 2 DL. Section 4 describes the mapping between ORM and OWL 2. Section 5 implements the mapping. Section 6 evaluates the mapping, and finally Section 7 concludes our work and states future work.

2. ORM Background

ORM is a fact-oriented modeling methodology independent from implementation-oriented procedures. This independence leads to a satisfactory modeling process [11]. ORM makes it easy to simplify the representative schema using either natural language or graphical notations to represent facts in their simple or elementary shapes. In addition, we can populate the diagrams by examples to measure the correctness of the design [7,8]. We have several tools based on ORM like Microsoft's Visio Modeler™, DogmaModeler and Norma. The knowledge of using ORM can be acquired easily and in a short period of time from non-IT specialists [11,13].
ORM can be fairly used to adopt the conceptual modeling techniques for building the needed ontology [6,13]. By using a graphical notation of ORM, we can express and treat many rules like mandatory, uniqueness, identity, exclusion, implications, frequency occurrences, subsetting, subtyping, equality, and others [7]. Many rules of ORM and their graphic representations are explained (see section 4).

3. OWL 2 DL

Ontology Web Language (OWL) is a knowledge representation language [20] used to publish and share ontologies on the Web and is endorsed by the W3C Consortium.
OWL 2 Web Ontology Language, is an ontology language for the semantic web (extended of OWL 1, empowered by new features and supported by several semantic reasoners such as RacerPro 2 and FaCT++). This ontology language includes formally defined meaning. On 27 October 2009, OWL 2 was recommended by W3C Consortium as a standard of ontology representation on the Web [17].Classes, properties, individuals, and data values are

Published As:
Rami Hodrob, Mustafa Jarrar: ORM to OWL 2 DL Mapping.. In proceedings of the International Conference on Intelligent Semantic Web – Applications and Services. Pages 131-137. ACM ISBN 9781450304757. June 2010.

```
<Declaration>
   <Class IRI="#Person"/>
…
<Declaration>
 <ObjectPropertyIRI="#DrivenBy"/>
   </Declaration>
<InverseObjectProperties>
    <ObjectProperty IRI="#DrivenBy"/>
    <ObjectProperty IRI="#Drives"/>
</InverseObjectProperties>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#Drives"/>
      <Class IRI="#Person"/>
 /ObjectPropertyDomain>
  <ObjectPropertyRange>
      <ObjectProperty IRI="#Drives"/>
      <Class IRI="#Vehicle"/>
  </ObjectPropertyRange>
```

provided by OWL 2 and stored semantically on the WEB. OWL 2 ontologies are primary exchanged as RDF documents, where these ontologies can be used with information written in RDF. OWL 2 elements are identified by Internationalized Resource Identifiers (IRIs). It extends OWL 1 which uses Uniform Resource Identifiers (URIs) [16,20]. Every IRI must be absolute to be published internationally. OWL 2 increases expressive language power for properties.

The new features of OWL 2 are i) syntactic sugar to make some statements easier. ii) new constructs that increase expressivity. iii) extended support for datatypes; iv) simple metamodeling capabilities; v) extended annotation capabilities. OWL 2 is serialized by XML to structurally specify it.

4. Mapping between ORM and OWL 2

Since we concentrate on the ability of expressivity and decidability for our mapping results(SHOIN achieves this Ability [11]), we will use SHOIN Description Logic (which is the most common in ontology engineering [11]) as a reference to map from ORM into OWL 2 DL. First, we formally formalize the ORM construct into SHOIN Description Logic and then we represent this model in OWL 2. Our scope of conversion is every construct of ORM .

4.1 Use Case

In order to recognize the ORM graphical notations, and Mapping between ORM, SHOIN and OWL 2 refer to Figure 1, and the explanation that follows.

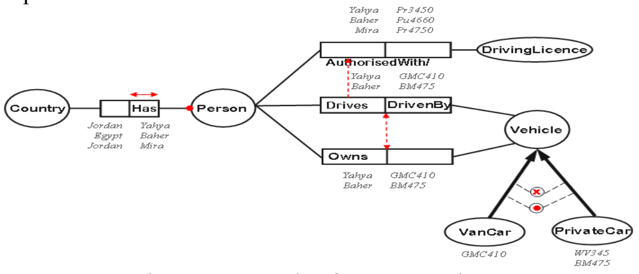


Figure 1. Example of an ORM Schema

In Figure 1, object-types are represented as ellipses, and relations as rectangles, where one or more role form each ORM relation. Binary relation (*Drives/DrivenBy)* in SHOIN is as *(Person* $\sqsubseteq$ $\forall$*Drives.Vehicle, Vehicle* $\sqsubseteq$ $\forall$*DrivenBy.Person, DrivenBy* $\sqsubseteq$ *Drives*$^{-}$*)*. This relation is represented in OWL 2 DL as shown in the OWL/XML syntax below. Object-type in ORM is declared as Class (*Person* and *Vehicle)* construct in OWL 2. Each role of the relation in ORM is declared as objectProperty (*Drives* and *DrivenBy)* construct in OWL 2.

In the following we explain each rule in ORM, its formalization in SHOIN and its representation in OWL 2 (rules represented in figure 1):

a. *Subsumption* is represented in SHOIN as (*VanCar* $\sqsubseteq$ *Vehicle, PrivateCar* $\sqsubseteq$ *Vehicle*). In OWL 2 subClassOf construct is used to represent this rule as

```
<SubClassOf>
     <Class IRI="#PrivateCar"/>
     <Class IRI="#Vehicle"/>
</SubClassOf>
<SubClassOf>
     <Class IRI="#VanCar"/>
     <Class IRI="#Vehicle"/>
 </SubClassOf>
```

b. *Mandatory* is depicted as a dot • on the line. In SHOIN is (*Person* $\sqsubseteq$ $\exists$*Has.Country*). In OWL 2 ObjectSomeValuesFrom construct which is equivalent to the extensional quantifier($\exists$) is used to represent Mandatory in ORM which is more elegant than using minCardinality construct to restrict the population of Person to at least have one Country as

```
<ObjectPropertyRange>
   <ObjectProperty IRI="#Has"/>
     <ObjectSomeValuesFrom>
       <ObjectProperty abbreviatedIRI="owl:topObjectProperty"/>
       <Class IRI="#Country"/>
      </ObjectSomeValuesFrom>
 </ObjectPropertyRange>
```

c. *Total constraint* is depicted as a dot (•) between the two subtypes. In SHOIN it is (*Vehicle* $\sqsubseteq$*VanCartPrivateCar*).OWL2, ObjectUnionOf construct is used to represent this rule.

d. *Exclusive Constraint* is depicted as ⊗ between the two subtypes. In SHOIN it is (*VanCar* $\sqcap$ *PrivateCar* $\equiv\bot$). In OWL 2 DisjointClasses expressions is used. We use DisjointUnion construct to map both Total Constraint and Exclusive as

```
</DisjointUnion>
     <Class IRI="#Vehicle"/>
     <Class IRI="#PrivateCar"/>
     <Class IRI="#VanCar"/>
</DisjointUnion>
```

e. *Subset Constraint* is depicted as an arrow → between the roles *Drives* and *AuthorizedWith*; which means that the object role Drives is a subset of object role AuthorizedWith. *(Drives.Vehicle*$\sqsubseteq$*AuthorizedWith.DrivingLicence*). In OWL 2, to represent this rule we consider Drives. Person as a class using equivelentClass construct in OWL2. This construct is subClassOf the equivelantClass (AuthorizedWith.Person) as shown below

f. *EqualityConstraint* is depicted as a double-headed arrow↔ (*Owns≡Drives*). Representation in OWL 2 is done using EquivalentObjectProperties as

```
<EquivalentObjectProperties>
    <ObjectProperty IRI="#Drives"/>
    <ObjectProperty IRI="#Owns"/>
</EquivalentObjectProperties>
```

g. *Role uniqueness* is depicted by an arrow ↔ spanning along single role of binary relation. In SHOIN (*Person* ⊑ ≤1*Has.Country).* In OWL 2 we use FunctionalObjectProperty (range is exactly one (for domain population of property)) which is more elegant than using maxCardinality (restricted by integer 1) construct. It is represented as

```
<FunctionalObjectProperty>
    <ObjectProperty IRI="#Has"/>
</FunctionalObjectProperty>
```

Verbalization of ORM rules

ORM diagrams can be read easily by domain experts, and rules can be automatically verbalized into pseudo natural language sentences as the following:

- Each Person Has at least one Country. *(Mandatory)*
- Each Vehicle can not be a VanCar and a PrivateCar at the same time. *(Exclusive)*
- Each Vehicle must be, at least, VanCar or PrivateCar. *(Totality)*
- If a Person Drives a Vehicle then that Person AuthorizedWith a DrivingLicence. *(Subset)*
- If a Person Owns a Vehicle this Person is also Drives that Vehicle, and vice versa. *(Equality)*
- Each Person must Has at most one Country. *(External uniqueness)*

```
<EquivalentClasses>
  <Class IRI="#AutorizedWith.Person"/>
<ObjectAllValuesFrom>
   <ObjectProperty IRI="#AuthorizedWith"/>
   <Class IRI="#Person"/>
</ObjectAllValuesFrom>
 </EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Drives.Person"/>
      <ObjectAllValuesFrom>
           <ObjectProperty IRI="#Drives"/>
           <Class IRI="#Person"/>
 </ObjectAllValuesFrom>
    </EquivalentClasses>
    <SubClassOf>
       <Class IRI="#Drives.Person"/>
       <Class IRI="#AutorizedWith.Pers>
</SubClassOf>
```

This verbalization simplifies the communication with non-IT specialists and allows them to better recognize, validate, or build ORM diagrams.

A complete list of ORM rules and mapping work in general cases is explained in the following.

4.2 Object-Types and relations:

4.2.1 Unary relationship

See the first column in Table 1.

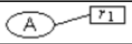
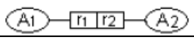
4.2.2 Binary relationship

The mapping is as stated in the example in section 4.1 (general case is shown in Table 1 right side).

4.2.3 N-ary relationships where n>2

It is not considered (not supported by SHOIN).

Table 1. Relationships (Unary and Binary) are represented in ORM, SHOIN and OWL 2 DL

ORM		
SHOIN	*A ⊑ ∀r1.Bolean*	*A1 ⊑ ∀r1.A2, A2 ⊑ ∀r2.A1, r2 ⊑ ∀r1⁻*

OWL 2	`<Declaration>`
	`<Class IRI="#A"/>`
	`</Declaration>`
	`<Declaration>`
	`<DataProperty IRI="#r1"/> </Declaration>`
	`<SubDataPropertyOf>`
	`<DataProperty IRI="#r1"/>`
	`<DataProperty abbreviatedIRI="owl:topDataProperty"/>`
	`</SubDataPropertyOf>`
	`<DataPropertyDomain>`
	`<DataProperty IRI="#r1"/>`
	`<Class IRI="#A"/>`
	`</DataPropertyDomain>`
	`<DataPropertyRange>`
	`<DataProperty IRI="#r1"/>`
	`<Datatype abbreviatedIRI="xsd:boolean"/>`
	`</DataPropertyRange>`
	…
		`<InverseObjectProperties>`
		`<ObjectProperty IRI="#r2"/>`
		`<ObjectProperty IRI="#r1"/>`
		`</InverseObjectProperties>`
		`<ObjectPropertyDomain>`
		`<ObjectProperty IRI="#r1"/>`
		`<Class IRI="#A1"/>`
		`</ObjectPropertyDomain>`
		`<ObjectPropertyDomain>`
		`<ObjectProperty IRI="#r2"/>`
		`<Class IRI="#A2"/>`
		`</ObjectPropertyDomain>`
		`<ObjectPropertyRange>`
		`<ObjectProperty IRI="#r1"/>`
		`<Class IRI="#A2"/>`
		`</ObjectPropertyRange>`
		`<ObjectPropertyRange>`
		`<ObjectProperty IRI="#r2"/>`
		`<Class IRI="#A1"/>`
		`</ObjectPropertyRange>`
		…

4.3 Subtypes

ORM uses proper subtype [8,11]. See the example in section 4.1 and Table 2 (first column).

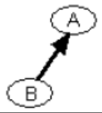
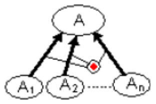
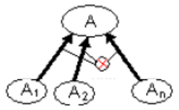
4.4 Total constraint

General case of mapping is shown in Table 2 (middle column).

4.5 Exclusive constraint

The general case of mapping is in Table 2 (last column).

Table 2. Subtype, Total Constraint and Exclusive Constraint (declaration of classes is not included)

ORM			
SHOIN	*B* ⊑ *A*	*A* v A1 t A2 … t An	*(Ai* ∪ *Aj* ≡ *⊥) for each i*∈*{1 …n-1},j*∈*{i+1 …n}*
OWL 2	… <SubClassOf> <Class IRI="#B"/> <Class IRI="#A"/> </SubClassOf> …	<SubClassOf> <ObjectUnionOf> <Class IRI="#A1"/> <Class IRI="#An"/> </ObjectUnionOf> <Class IRI="#A"/> </SubClassOf>	… <DisjointClasses> <Class IRI="#A1"/> <Class IRI="#A2"/> … <Class IRI="#An"/> </DisjointClasses>

4.6 Mandatory Constraints

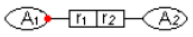
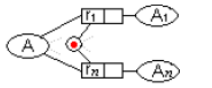
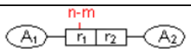
4.6.1 Role mandatory

The mapping is as stated in the example in section 4.1 (see the first column in Table 3).

4.6.2 Disjunctive Mandatory

The disjunctive mandatory constraint is as stated in middle column of table 3, means that each instance of object-type A must play the role of at least one of the constraints role r1….rn. The representation of this in OWL2 is shown in Table 3 (middle column).

Table 3 Mandatory Constraints and Role Frequency Constraint (Classes and ObjectProperties are not declared)

ORM			
SHOIN	*A1* ⊑ ∃*r1.A2*	*A* v ∃*r1.A1* … t ∃*rn.An*	*A1* v ≥ n, ≤ *m r1.A2* t ⊥
OWL 2	</ObjectPropertyRange> <ObjectPropertyRange> <ObjectProperty IRI="#r1"/> <ObjectSomeValuesFrom> <ObjectProperty IRI="#r1"/> <Class IRI="#A2"/> </ObjectSomeValuesFrom> </ObjectPropertyRange>	<ObjectPropertyRange> <ObjectProperty IRI="#r1"/> <ObjectSomeValuesFrom> <ObjectProperty IRI="#r1"/> <Class IRI="#A1"/> </ObjectSomeValuesFrom> </ObjectPropertyRange> …	<ObjectPropertyRange> <ObjectProperty IRI="#r1"/> <ObjectMinCardinality cardinality="n"> <ObjectProperty IRI="#r1"/> <Class IRI="#A2"/> </ObjectMinCardinality> … <ObjectMaxCardinality cardinality="m"> …

4.7 Role Uniqueness

Refer to Subsection 4.1.

4.8 Frequency Constraints

4.8.1 Role Frequency Constraint

Role Frequency in ORM means that role r1 is played by the object A2 number of occurrences (see Table 3 last column).

4.8.2 Multiple-role Frequency Constraint

Can not be formalized in description logic [4] and so it is not considered in OWL 2.

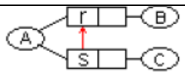
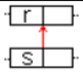
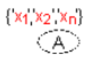
4.9 Value Constraints

The value constraint in ORM points to the possible set of values that an object-type can be populated with (Table 4 last column).

4.10 Subset Constraints

Stated in the example in section 4.1 (Table 4 (first two columns)).

Table 4 Subset Constraint (Role and Binary) and Value Constraint (String Type)

ORM			
SHOIN	*s.C ⊑ r.B*	*s ⊑ r*	*A1 ⊑ STRING A ≡ {x1,…,xn}*
OWL 2	<EquivalentClasses> <Class IRI="#r.A"/> <ObjectAllValuesFrom> <ObjectProperty IRI="#r"/> <Class IRI="#A"/> </ObjectAllValuesFrom> </EquivalentClasses> <EquivalentClasses> <Class IRI="#s.A"/> <ObjectAllValuesFrom> <ObjectProperty IRI="#s"/> <Class IRI="#A"/> </ObjectAllValuesFrom> </EquivalentClasses> <SubClassOf> <Class IRI="#s.A"/> <Class IRI="#r.A"/> </SubClassOf>	… <SubObjectPropertyOf> <ObjectProperty IRI="#s"/> <ObjectProperty IRI="#r"/> </SubObjectPropertyOf> …	<EquivalentClasses> <Class IRI="#A"/> <DataAllValuesFrom> <DataProperty abbreviatedIRI="owl:topDataProperty"/> <Datatype abbreviatedIRI="xsd:string"/> </DataAllValuesFrom> </EquivalentClasses> <EquivalentClasses> <Class IRI="#A"/> <ObjectOneOf> <NamedIndividual IRI="#X1"/> … <NamedIndividual IRI="#Xn"/> </ObjectOneOf> </EquivalentClasses>

4.11 Equality Constraint

It is similar to subset constraint. In OWL 2, we use (EquivalentObjectProperties) construct to represent it.

4.12 Exclusion Constraint

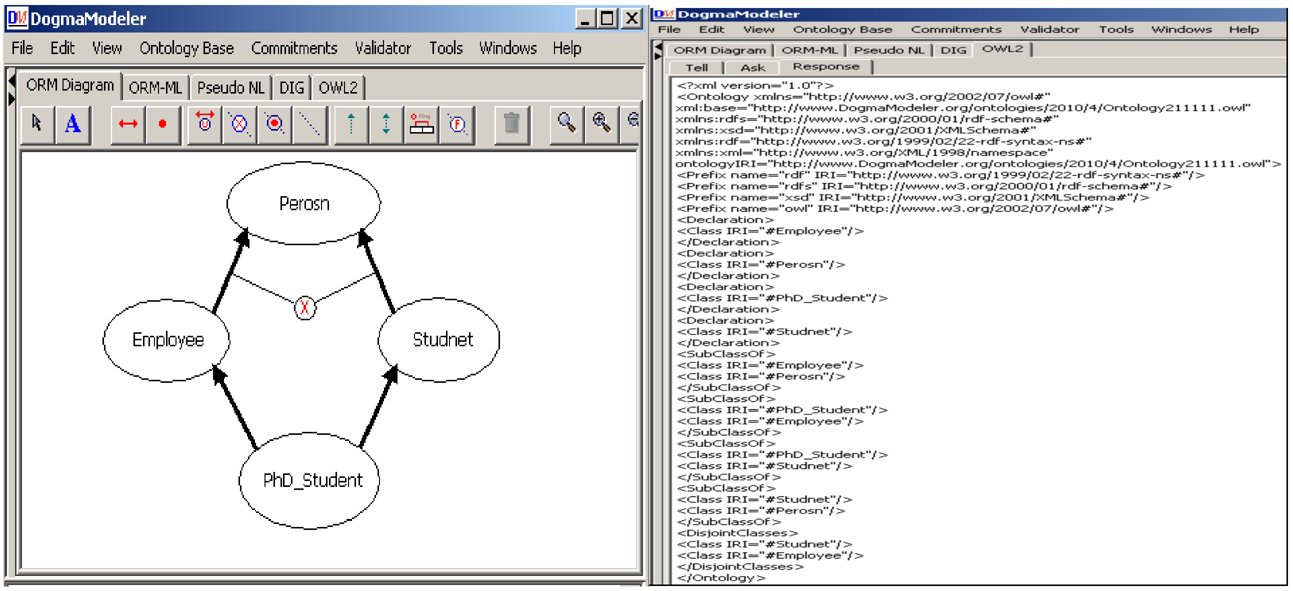
It is similar to subset constraint. In OWL 2, we use (DisjointObjectProperties) construct to represent it.

4.13 Ring Constraints

OWL 2 supports Reflexive, Irreflexive, and Asymmetric object properties as new features in addition to Symmetric and Transitive that are supported by OWL 1 (equivalent constructs are used in ORM).

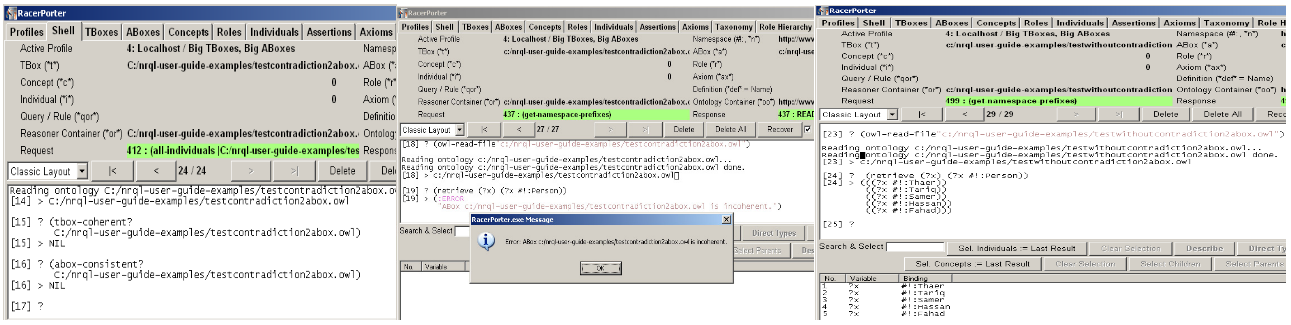
5. Implementation

We implement our mapping using DogmaModeler. DogmaModeler is a modeling tool used to represent and reason for ontology and other related applications based on ORM. We have extend DogmaModeler (Java is used as a programming language for coding) to automatically map ORM into OWL 2 DL constructs depending on ORM markup language [3,12] (which is automatically generated according to equivalent ORM graphical notations). Figure 2 shows a snap shot of DogmaModeler outputs, where the left screen shows the ORM graphical notation containing subtypes and exclusive constraint. The right screen shows a complete OWL 2 file output that represents the ORM graphical notation.


DogmaModeler
File Edit View Ontology Base Commitments Validator Tools Windows Help
ORM Diagram ORM-ML Pseudo NL DIG OWL2
Perosn
Employee
Studnet
PhD_Student
DogmaModeler
File Edit View Ontology Base Commitments Validator Tools Windows Help
ORM Diagram ORM-ML Pseudo NL DIG OWL2
Tell Ask Response

```
<?xml version="1.0"?>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
xml:base="http://www.DogmaModeler.org/ontologies/2010/4/Ontology211111.owl"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:xml="http://www.w3.org/XML/1998/namespace"
ontologyIRI="http://www.DogmaModeler.org/ontologies/2010/4/Ontology211111.owl">
<Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
<Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
<Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
<Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
<Declaration>
<Class IRI="#Employee"/>
</Declaration>
<Declaration>
<Class IRI="#Perosn"/>
</Declaration>
<Declaration>
<Class IRI="#PhD_Student"/>
</Declaration>
<Declaration>
<Class IRI="#Studnet"/>
</Declaration>
<SubClassOf>
<Class IRI="#Employee"/>
<Class IRI="#Perosn"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#PhD_Student"/>
<Class IRI="#Employee"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#PhD_Student"/>
<Class IRI="#Studnet"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Studnet"/>
<Class IRI="#Perosn"/>
</SubClassOf>
<DisjointClasses>
<Class IRI="#Studnet"/>
<Class IRI="#Employee"/>
</DisjointClasses>
</Ontology>
```

Figure 2. Implemented example using DogmaModeler (ORM graphical notation and OWL 2 DL).



```
Active Profile            4: Localhost / Big TBoxes, Big ABoxes
TBox (*t*)                C:/nrql-user-guide-examples/testcontradiction2abox.
Concept (*c*)                                    0
Individual (*i*)                                 0
Query / Rule (*qor*)
Reasoner Container (*or*) C:/nrql-user-guide-examples/testcontradiction2abox.
Request                   412 : (all-individuals |C:/nrql-user-guide-examples/tes
Classic Layout   24 / 24

Reading ontology C:/nrql-user-guide-examples/testcontradiction2abox.ow
[14] > C:/nrql-user-guide-examples/testcontradiction2abox.owl

[15] ? (tbox-coherent?
        C:/nrql-user-guide-examples/testcontradiction2abox.owl)
[15] > NIL

[16] ? (abox-consistent?
        C:/nrql-user-guide-examples/testcontradiction2abox.owl)
[16] > NIL

[17] ?
```

```
Active Profile            4: Localhost / Big TBoxes, Big ABoxes     Namespace (#!:, *n*)     http://www
TBox (*t*)                c:/nrql-user-guide-examples/testcontradiction2abox.  ABox (*a*)  c:/nrql-use
Concept (*c*)                                    0                  Role (*r*)
Individual (*i*)                                 0                  Axiom (*ax*)
Query / Rule (*qor*)                                                Definition (*def* = Name)
Reasoner Container (*or*) c:/nrql-user-guide-examples/testcontradiction2abox.  Ontology Container (*oo*) http://www
Request                   437 : (get-namespace-prefixes)            Response                 437 : READ
Classic Layout   27 / 27

[18] ? (owl-read-file"c:/nrql-user-guide-examples/testcontradiction2abox.owl")

Reading ontology c:/nrql-user-guide-examples/testcontradiction2abox.owl...
Reading ontology c:/nrql-user-guide-examples/testcontradiction2abox.owl done.
[18] > c:/nrql-user-guide-examples/testcontradiction2abox.owl

[19] ? (retrieve (?x) (?x #!:Person))
[19] > (:ERROR
         "ABox c:/nrql-user-guide-examples/testcontradiction2abox.owl is incoherent.")

RacerPorter.exe Message
Error: ABox c:/nrql-user-guide-examples/testcontradiction2abox.owl is incoherent.
```

```
Active Profile            4: Localhost / Big TBoxes, Big ABoxes     Namespace (#!:, *n*)
TBox (*t*)                c:/nrql-user-guide-examples/testwithoutcontradiction  ABox (*a*)
Concept (*c*)                                    0                  Role (*r*)
Individual (*i*)                                 0                  Axiom (*ax*)
Query / Rule (*qor*)                                                Definition (*def* = Name)
Reasoner Container (*or*) c:/nrql-user-guide-examples/testwithoutcontradiction  Ontology Container (*oo*)
Request                   499 : (get-namespace-prefixes)            Response
Classic Layout   29 / 29

[23] ? (owl-read-file"c:/nrql-user-guide-examples/testwithoutcontradiction2abox.owl")

Reading ontology c:/nrql-user-guide-examples/testwithoutcontradiction2abox.owl...
Reading ontology c:/nrql-user-guide-examples/testwithoutcontradiction2abox.owl done.
[23] > c:/nrql-user-guide-examples/testwithoutcontradiction2abox.owl

[24] ?  (retrieve (?x) (?x #!:Person))
[24] > (((?x #!:Thaer))
        ((?x #!:Tariq))
        ((?x #!:Samer))
        ((?x #!:Hassan))
        ((?x #!:Fahad)))

[25] ?

No.  Variable  Binding
1    ?x        #!:Thaer
2    ?x        #!:Tariq
3    ?x        #!:Samer
4    ?x        #!:Hassan
5    ?x        #!:Fahad
```

Figure 3. RacerPro 2 outputs for consistent, coherent, and instance retrieval checks.

6. Evaluation

For the evaluation part, every construct of OWL 2 mapped from ORM is loaded as a complete file to the RacerPro 2 [18] that supports OWL 2. We have many checks (such as consistency and coherent checks) concerning reasoning techniques [15] used to validate the ontology represented in OWL 2. RacerPro 2 checks the coherence of TBos. If it is coherent, it will give t (which means true). If not, it will give NIL. Another check done by RacerPro is the consistency of ABox to check if it is as a model consistent with TBox. If so, the reasoner will give t (true). Another way of reasoning using RacerPro2 is creating queries using the New RacerPro Query Language-nRQL[19], by populating the TBox of knowledge base. Then, we check the consistency of knowledge base. We load the OWL 2 file (shown in Figure 2 left side) into the RacerPro 2. When we check the coherence of TBox, it gives us NIL (see Figure 3 left side) because of the contradiction between Exclusive constraint and PhDStudent subtype Class. When we insert individuals into the knowledge base for classes Student, Employee and PhD_Student, and checked the consistency of ABox with TBox, it gives us NIL because of the contradiction mentioned above. After populating the classes Student, Employee and PhD Student and applying nRQL for individual retrieval, RacerPro 2 gives us that the knowledge base was incoherent and there is no valid model for the class Person (Fig 3 middle screen). Where we exclude the exclusive constraint and check the consistency of ABox, it gives us a valid model for Person, and ABox is consistent (see figure 3 right side) with the TBox (axioms represented in OWL 2 (see figure 2 right side)). This output of check (instance retrieval) proves the correctness of mapping Exclusive construct of ORM (represented in graphical notation (figure 2 left side) into OWL 2 construct (DisjointClasses which is represented in Figure 2 right side).

7. Conclusion and Future Work

The mapping and automation of this mapping from ORM into OWL 2 are the main theme of this paper. We do map twenty two (out of twenty nine) ORM constructs. Where these 22 constructs represent the most commonly used constructs in ORM. At the same time, those constructs are supported by SHOIN Description Logic; which means the OWL 2 output we have mapped characterized by its ability of decidability. Through the evaluation process, we illustrated the correctness of our mapping. The importance is not in the mapping itself, but in the outcome because of the existence of a large number of applications that depend on it. OWL 2 new features inspired the mapping of some ORM constructs that were not supported by OWL 1 such as DisjointUnion, Ring Constraints (Reflexive, Irreflexive, and Asymmetric Object Properties) and others.

Some of the OWL 2 constructs are not supported by ORM such as equivalent class, etc. We plan to work on that in the future.

References:

[1] Cuyler D., and Halpin T. 2005. Two Meta-Models for Object-Role Modeling, Information Modeling Methods and Methodologies, eds . Idea Publishing Group, Hershey PA, USA , 17-42.

[2] Cranefield P. S., Hart L., Dutra M., Baclawski K., Kokar M., and Smith J. 2002. Uml for ontology development. Knowl. Eng. Rev., 17(1):61–64,

[3] Demey J., Jarrar M., and Meersman R. June, 2002. A Markup Language for ORM Business Rules. (RuleML 2002). Volume 60 of CEUR Workshop Proceedings, 107-128, CEUR-WS.org..

[4] DogmaModeler:www.starlab.vub.ac.be/research/dogma/dogmamodeler/dm.ht.

[5] Guarino N. 1998. Formal ontologies and information systems. Proceedings of FOIS.. Amsterdam, IOS Press (June, 1998), 3-15.

[6] Guizzardi G., and Halpin T. A. 2008. Ontological foundations for conceptual modelling. Applied Ontology 3(1-2): 1-12.

[7] Halpin T. 2004. Object-Role Modeling: an overview. Microsoft Corporation.

[8] Halpin T. 2001. Information Modeling and Relational Databases From Conceptual Analysis to Logical Design. Morgan Kufmann. www.w3.org/2009/pdf/REC-owl2-overview-20091027.pdf

[9] Halpin T. 2004. Business Rule Verbalization, Information Systems Technology and its Applications. Proceedings of ISTA-2004, vol. P-48, 39-52.

[10] Jarrar M. 2007. Towards automated reasoning on orm schemes. In Proceedings of the 26th International Conference on Conceptual Modeling (ER 2007). Springer,

[11] Jarrar M. 2007. Mapping ORM into the SHOIN/OWL Description Logic – Towards a Methodological and Expressive Graphical Notation for Ontology Engineering. OTM Workshops(ORM'07),LNCS480 Springer. http://www.jarrar.info/Publications/.

[12] Jarrar M. May, 2005. Towards Methodological Principles for Ontology Engineering. PhD thesis,Vrije Universiteit Brussel, Brussels, Belgium.

[13] Jarrar M., Demey J., and Meersman R. 2003. On using conceptual data modeling for ontology engineering. Journal on Data Semantics (October, 2003).

[14] McGuinness D. L. McGuinness, and Harmelen F. V. 2004. OWL Web Ontology Language Overview. W3C Recommendation.(Feb.,2004).http://www.w3.org/TR/2004/REC-owl-features-20040210.

[15] Nardi D., and Brachman R. J. An Introduction to Description Logics. In the Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2002. http://www.inf.unibz.it/~franconi/dl/course/dlhb/dlhb-01.pdf

[16] http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/

[17] (www.w3.org/TR/owl2-profiles/)

[18] http://www.racersystems.com/products/racerpro/preview/overview.phtml

[19] http://www.racersystems.com/products/racerpro/users-guide-1-9-2-beta.pdf

[20] Smith M. K., Welty C., and McGuinness D. L. February, 2004. OWL Web Ontology Language Guide.