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 ModelerTM, 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
The OWL/XML syntax below. Object-type in ORM is declared as Drives

Binary relation (as rectangles, where one or more role form each ORM relation. 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 \( \sqsubseteq \forall \text{Drives.Vehicle} \), Vehicle \( \sqsubseteq \forall \text{DrivenBy.Person} \), DrivenBy \( \sqsubseteq \text{Drives} \)). This relation is represented in OWL 2 DL as shown in the OWL/XML syntax below. Object-type in ORM is declared as

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

This relation is represented in OWL 2 DL as shown in the OWL/XML syntax below. Object-type in ORM is declared as

```
<Class IRI="#Vehicle"/>
<Class IRI="#Person"/>
<ObjectProperty IRI="#Drives"/>
<ObjectPropertyRange>
  <ObjectProperty IRI="#Drives"/>
  <Class IRI="#Vehicle"/>
</ObjectPropertyRange>
```

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 \((\text{VanCar} \sqsubseteq \text{Vehicle})\). In OWL 2 `SubClassOf` construct is used to represent this rule as

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

b. **Mandatory** is depicted as a dot • on the line. In SHOIN it is \((\text{Person} \sqsubseteq \exists \text{Has.Country})\). In OWL 2 `ObjectSomeValuesFrom` construct which is equivalent to the extensional quantifier(3) 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"/>
</ObjectPropertyRange>
```

c. **Total constraint** is depicted as a dot (●) between the two subtypes. In SHOIN it is \((\text{Vehicle} \sqsubseteq \forall \text{VanCar.PrivateCar})\). OWL2, `ObjectUnionOf` construct is used to represent this rule.

d. **Exclusive Constraint** is depicted as ◈ between the two subtypes. In SHOIN it is \((\text{VanCar} \sqsubseteq \text{PrivateCar})\). In OWL 2 `DisjointClasses` expresssions is used. We use `DisjointUnion` construct to map both Total Constraint and Exclusive as

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

e. **Subset Constraint** is depicted as an arrow \(\rightarrow\) between the roles Drives and AuthorizedWith; which means that the object role Drives is a subset of object role AuthorizedWith.

```
(\text{Drives.Vehicle} \subseteq \text{AuthorizedWith.DrivingLicence})
```

In OWL 2, to represent this rule we consider Drives. Person as a class using equivalentClass construct in OWL2. This construct is subClassOf the equivelantClass (AuthorizedWith Person) as shown below
1. EqualityConstraint is depicted as a double-headed arrow ↔ (Owns=Drives). Representation in OWL 2 is done using EquivalentObjectProperties as

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

2. Role uniqueness is depicted by an arrow ↔ spanning along single role of binary relation. In SHOIN (Person ⊆ ≤1Has.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)

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

<table>
<thead>
<tr>
<th>ORM</th>
<th>A ⊆ r1.Bolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHOIN</td>
<td>A ⊆ ∀r1.A</td>
</tr>
<tr>
<td>OWL 2</td>
<td>A ⊆ ∀r1.A</td>
</tr>
</tbody>
</table>

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).

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)

<table>
<thead>
<tr>
<th>ORM</th>
<th>SHOIN</th>
<th>OWL 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$B \subseteq A$</td>
<td>$A \lor A_1 \land A_2 \ldots \land A_n$ ((A_i \lor A_j = \perp) for each (i \in {1 \ldots n-1}, j \in {i+1 \ldots n}))</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>ORM</th>
<th>SHOIN</th>
<th>OWL 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_1 \subseteq A_2</td>
<td>$A \lor \exists r_1.A_1 \ldots \land \exists m.A_n$</td>
<td>$A_1 \lor A_2 \land \exists m.A_n$</td>
</tr>
<tr>
<td>OWL 2</td>
<td>$\langle \text{ObjectPropertyRange}\rangle$</td>
<td>$\langle \text{ObjectPropertyRange}\rangle$</td>
</tr>
<tr>
<td></td>
<td>$\langle \text{ObjectPropertyRange}\rangle$</td>
<td>$\langle \text{ObjectPropertyRange}\rangle$</td>
</tr>
<tr>
<td></td>
<td>$\langle \text{ObjectProperty IRI=&quot;#r1&quot;}\rangle$</td>
<td>$\langle \text{ObjectProperty IRI=&quot;#r1&quot;}\rangle$</td>
</tr>
<tr>
<td></td>
<td>$\langle \text{ObjectSomeValuesFrom}\rangle$</td>
<td>$\langle \text{ObjectSomeValuesFrom}\rangle$</td>
</tr>
<tr>
<td></td>
<td>$\langle \text{ObjectProperty IRI=&quot;#r1&quot;}\rangle$</td>
<td>$\langle \text{ObjectProperty IRI=&quot;#r1&quot;}\rangle$</td>
</tr>
<tr>
<td></td>
<td>$\langle \text{ObjectSomeValuesFrom}\rangle$</td>
<td>$\langle \text{ObjectSomeValuesFrom}\rangle$</td>
</tr>
<tr>
<td></td>
<td>$\langle \text{ObjectPropertyRange}\rangle$</td>
<td>$\langle \text{ObjectPropertyRange}\rangle$</td>
</tr>
<tr>
<td></td>
<td>$\langle \text{ObjectProperty IRI=&quot;#r1&quot;}\rangle$</td>
<td>$\langle \text{ObjectProperty IRI=&quot;#r1&quot;}\rangle$</td>
</tr>
<tr>
<td></td>
<td>$\langle \text{ObjectSomeValuesFrom}\rangle$</td>
<td>$\langle \text{ObjectSomeValuesFrom}\rangle$</td>
</tr>
<tr>
<td></td>
<td>$\langle \text{ObjectPropertyRange}\rangle$</td>
<td>$\langle \text{ObjectPropertyRange}\rangle$</td>
</tr>
<tr>
<td></td>
<td>$\langle \text{ObjectProperty IRI=&quot;#r1&quot;}\rangle$</td>
<td>$\langle \text{ObjectProperty IRI=&quot;#r1&quot;}\rangle$</td>
</tr>
</tbody>
</table>

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)
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.

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

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 TBox. 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


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

