Representation and Conformance of UML models containing ordered properties using OWL2

Ali Hanzala Khan, Espen Suenson, and Ivan Porres

TUCS Turku Centre for Computer Science
D. of Information Technologies, Åbo Akademi University
Joukahaisenkatu 3-5, FI-20520 Turku, Finland
name.surname@abo.fi

Abstract. In this article we show how to represent UML models depicting ordered properties using OWL2, and how to reason about model conformance using OWL2 reasoners. Our translation from UML models to OWL2 is driven by three important forces. First, we want to maintain the close-world assumption about UML models. Second, we want to preserve structural model information of ordered properties. Finally, model conformance is defined solely by OWL2 axioms so that reasoning can be done by using existing and future OWL2 reasoner developed by others. We have implemented the translation as an automatic model transformation tool. The model transformation tool takes as input a UML object model and its class model and produces an ontology that can be processed by an OWL2 reasoner to reveal if the object model elements conform to their class model or not.
Keywords: Model Validation, OWL2, Ordering, Reasoning.

1 Introduction

Model Driven Engineering (MDE) [9] advocates the use of models to represent the most relevant design decisions in a software development process. Each software model is described using a particular modeling language, such as the Unified Modeling Language (UML) [16] or a domain specific modeling language (DSML) [6]. A complex development project involves the creation of many different models often using different modeling languages, and this raises the question if each model conforms to its metamodel or not.

In this article, we address the problem of the conformance of a UML object model against a UML class model containing ordered properties. In our context, conformance means that given a UML class model containing classes and associations and a UML object model containing objects and links, we want to know if each object is a proper instance of a class and each link is an instance of an association depicted in the class model.

In a UML class model an ordered property represents an ordered set of facts. It is represented diagrammatically by using a normal association and labeled as

”ordered”, whereas in a UML object model an instance of an ordered property is represented with an indexed link, and requires that the index is always unique and in order. The UML Superstructure metamodel [16] contains more that 50 ordered properties. An ordered property is used for example to hold the sequence of parameter in an operation. An example of class and object models that contain an ordered property is shown in Figure 1.

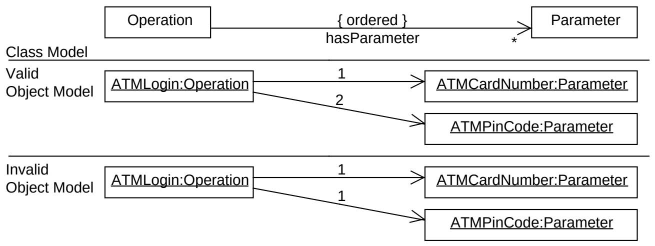

Fig. 1. Top: UML class model depicting ATM Machine login operation by using ordered property, Middle: Consistent UML object model, Bottom: Inconsistent UML object model due to the non-unique index link.

2 Overview

In order to reason about model conformance we need to have a formal definition of the UML. In this article we have chosen Web Ontology Language version 2 for Description Logic (OWL2 DL) [21] to formalize the UML class and object modeling concepts. There are a number of reasons behind the selection of OWL2 DL. Firstly, OWL2 DL is the subset of OWL2 that it is decidable. Secondly, by using OWL2 DL we will be able to use existing OWL2 reasoners for model conformance. Finally, OWL2 DL already provides constructs to represent many of the concepts of the UML such as classes, associations, individuals, and links in a rather straightforward way.

Still, a translation of UML models to OWL2 presents several challenges. Unfortunately, OWL2 DL does not provide any constructions to represent ordered facts in UML models directly. Also, OWL2 uses open-world assumption, whereas, UML operates under the closed-world assumption [10], where complete knowledge of the domain is assumed to be provided in a model. We assume that all existing classes and objects are known and depicted explicitly in the models.

The details about representing ordered facts and the closed-world environment in OWL2 DL are the main contribution of the article and will be discussed in Sections 4 and 5 in detail.

2.1 Automatic Model Conformance

To tackle the problem of model conformance, we propose to use a tool to first translate UML models into OWL2 DL, and then check the translated models for consistency using an OWL2 reasoner. An overview of the process can be seen in Figure 2. The translator will take UML class and object models as an input in form of XMI [15] and produce formally defined ontology in OWL2 format as an output. The translator contains the translations of UML concepts into OWL2 axioms in form of MOFScript [4]. The detail about the translation of a UML class and object modeling concepts into OWL2 will be discussed in Section 4 and 5 respectively, and the detail about the translation process will be discussed in Section 6. Furthermore, the ontology generated by the translator, contains

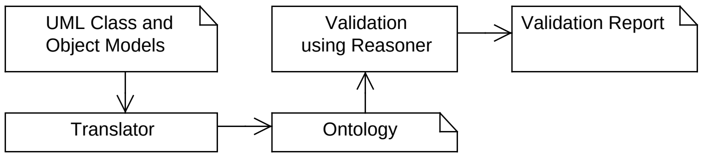

Fig. 2. Automatic object and class model conformance process.

the translated object model and a class model in form of OWL2 DL, will further be validated by using an OWL2 reasoner.

– If a generated ontology is consistent, it means that the object model conforms to all constraints of the class model interpreted according to the semantics that we have given in our translation.
– If a generated ontology is inconsistent, it means that the object model does not conform to the constraints given in the class model.

Moreover, in our previous work [7] and [11] we have described an automatic validation process of a UML class diagrams using an OWL2 reasoner. In this article, firstly we show the translations that map a UML class model and an object model into OWL2. Secondly, we will discuss how to enforce a UML close-world assumption in OWL2, with regard to UML constrain like ordering. Furthermore, we will discuss the implementation of our proposed model conformance process by using a MOFScript translation language and OWL2 reasoner, and lastly we will validate our approach with the help of a test case.

2.2 Related Work

The use of ontology languages and description logic in the context of model validation has been proposed in the past by different authors [19, 17, 12, 18, 22, 5, 1]. However, to our knowledge, none of them has addressed the reasoning of ordered properties. Furthermore, the validation of UML models using OCL has been discussed by [8, 13]. Nevertheless, these works focus on the problem of class model satisfiability, e.g. A class model can generate consistent object models or not.

3 Definition of UML Models

We consider a UML class model as a set of classes and their relationships in form of generalization and associations, as shown on top of Figure 3. Whereas, a UML object model consist of objects and links, where objects are the instances of a class and links are the instances of an association, as shown at the bottom of Figure 3. In this section, we will give a formal definition of the UML class and object modeling concepts that we require for the reasoning of ordered properties. The definition is given in terms of predicate logic. In this section we will also motivate our choice of features that are included in the definition.

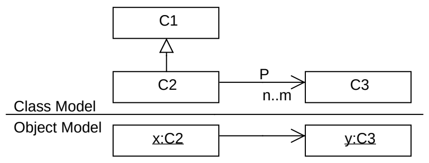

Fig. 3. Top: A UML class model depicting class hierarchy and association. Bottom: A UML object model.

3.1 Class and Class hierarchy

A UML class represents a set of objects that have the same characteristics [14]. A UML class C is defined as a unary predicate C in predicate logic. In UML, any two classes that do not share subclasses are considered as disjoint classes, the basic restriction on any two disjoint classes $C1$ and $C2$ that does not share any object x:

$$\forall x. \neg (C1(x) \wedge C2(x)) \tag{1}$$

3.2 Objects

An instance of a class is called an object. In UML, every object in an object model must belong to a specific class in a class model. An object x in an object model belongs to a class C in a class model is defined in predicate logic as:

$$C(x) \tag{2}$$

3.3 UML Associations

A UML association defines a relationship between two classes; every association is connected with a domain class and a range class. An association P is modeled by a binary predicate P in predicate logic. If $C1$ is the domain and $C2$ is the range class, and x, y are the objects of a domain and a range class, the basic restriction on P is:

$$\forall x, y. P(x, y) \rightarrow C1(x) \wedge C2(y) \tag{3}$$

Furthermore, UML associations can be specified with minimum and maximum multiplicity. This means that there are cardinality restrictions on how many objects from the range class can link to an object of the domain class. If the minimum multiplicity is n and the maximum multiplicity is m on association P then:

$$\forall x. C(x) \rightarrow n \leq \#\{y \mid P(x, y)\} \leq m \tag{4}$$

Where n, m are non-negative integers and $\#X$ is the cardinality of X.

3.4 Links

A link is an instance of an association. A link of an association P connecting objects x and y is represented in predicate logic as:

$$P(x, y) \tag{5}$$

If there exists no link between the objects of a domain class and a range class of a UML association, we will explicitly mention the negative assertion between those objects, and in predicate logic we can define the absence of a link between the objects x and y of an association P as:

$$\neg P(x, y) \tag{6}$$

The negative assertion is only required for those objects whose specific classes are related with each other by an association but their objects are not connected with the links.

3.5 Ordering

In UML ordering, the links of an ordered property are labeled with a unique numbered index, and it is required that the indexes are in order. An ordering would be used for example to preserve a sequence of a parameter in a function. A link of an ordered property P connecting the object x of source class and the object y of target class of an ordered property P having a label i is represented in predicate logic as:

$$P(x, y, i) \tag{7}$$

Furthermore, all links of an ordered property having identical index i are required to have an identical source and target, in predicate logic it is represented as:

$$\forall i \in N \; \forall x, y, a, b. P(x, a, i) \wedge P(y, b, i) \rightarrow (x = y) \wedge (a = b) \tag{8}$$

Where, x, y are the objects of a domain class and a, b are the objects of a range class of an ordered property P.

4 UML Class Model Formalization in OWL2

OWL2 DL is a decidable fragment of OWL2, and it is based on Description logic (DL). In this section, we will show the translation of UML Class modeling concepts discussed in Section 3 in to OWL2 DL.

4.1 UML Class

A class represents a collection of objects which share same features, constraints and definition. A UML Class C in the class model is translated in OWL2 as:

```
Declaration( Class( C ) )
```

Moreover, in UML one object in an object model can only belong to one class in a class model, whereas in an open-world assumption of OWL2 one individual can belong to many classes, except those classes that are declared as disjoint. It is therefore necessary to explicitly declare all classes which are not a superclass, a subclass, or sharing any of the subclasses as disjoint. In OWL2 disjointness (1) among classes is represented as:

```
DisjointClasses( C1 C2 ) )
```

4.2 UML Association

A UML association defines a relationship between two classes. It can be labeled as ordered. A UML association (3) is represented in OWL2 as ObjectProperty, an association P from a class C1 to a class C2 is represented in OWL2 as:

1. Declare a UML association P as an *objectproperty*:

   ```
   Declaration( ObjectProperty( P ) )
   ```
2. Assign a domain *C*1 to the property *P*:

   ```
   ObjectPropertyDomain( P C1 )
   ```
3. Assign a range *C*2 to the property *P*:

   ```
   ObjectPropertyRange( P C2 )
   ```

Associations Multiplicity. A UML association defines the multiplicity (4) in a non negative integer, which describes the number of allowable instances of a range class. We map the multiplicity of a UML association into OWL2, by defining the domain class of an association as a subclass of a set of classes, which relates with the same property and the given cardinality.

The UML association *P* from class *C*1 and *C*2 has a multiplicity constraint of n..m is represented in OWL2 as:

1. Define a minimum cardinality:

   ```
   SubClassOf( C1 ObjectMinCardinality( n P ) )
   ```
2. Define a maximum cardinality:

   ```
   SubClassOf( C1 ObjectMaxCardinality( m P ) )
   ```

5 Object Model Representation

An object model consists of objects and links, where each object is an instance of a UML class defined in the class model and each link is an instance of an association. A consistent object model must not contain a definition of a new class or an association, and all objects that exist in the object model must be an instance of the classes that exist in the class model. Similarly, all links present in the object model, must be an instance of an association that exists in the class model. If any of the object or link, present in an object model, is not an instance of an association or a class that exist in the class model, then that object model will be considered inconsistent.

In this section we show how to translate UML object modeling concepts discussed in Section 3 into OWL2 DL, while discussing the translation, we will also discuss that how to apply close-world restrictions in OWL2 at model level.

5.1 Objects

Every object exist in an object model must belong to a specific class in a class model. In OWL2 the UML object (2) is represented as a *classassertion* and called *individual*. A UML object x of class C is translated in OWL2 as:

```
ClassAssertion( C x )
```

Furthermore, every object in a UML object model is by default different from another. Whereas, in OWL2 due to the open-world assumption, we need to explicitly mention that all individuals are different from each other. For example: for objects $x1, .., xn$ in an object model, we use OWL2 axiom:

```
DifferentIndividuals(x1..xn)
```

5.2 Links

A UML association assertion between the objects in an object model is called *link* (5). A link in OWL2 is represented as a *propertyassertion*. The link of an association P between the objects x1 and x2 in an object model is represented in OWL2 as:

```
ObjectPropertyAssertion( P x1 x2 )
```

Moreover, due to the open-world assumption of OWL2, for a reasoner to be able to detect a violation of a minimum multiplicity constraint, we need to provide a definitive knowledge about the links, connecting or not connecting the individuals of a domain class and a range class of an association. Therefore, if there is no link between the objects of a domain class and a range class of a UML association, we need to explicitly declare that there is no connection between the individuals. The knowledge about the non-existence of link between individuals is called a *negativeassertion*. The negative assertion of an association P between the objects x1 and x2 in OWL2 is written as:

```
NegativeObjectPropertyAssertion( P x1 x2 )
```

A negative assertion is required when there exist an association between the classes but two specific individuals are not connected with a link.

5.3 Ordered Properties

The UML ordered property is translated in OWL2 as normal objectproperty. The translation of basic constraints like domain, range and multiplicity is also same as mention in Section 4.2. For example the translation of an ordered property P depicted on top of Figure 4 is as follows.

```
Declaration( ObjectProperty( P ) )
ObjectPropertyDomain( P C1 )
ObjectPropertyRange( P C2 )
SubClassOf( C1 ObjectMinCardinality( n P ) )
SubClassOf( C1 ObjectMaxCardinality( m P ) )
```

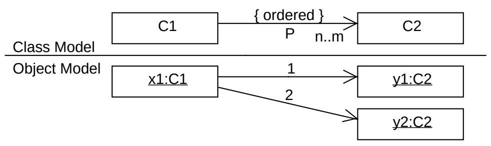

Fig. 4. Top: A UML class model depicting ordered property. Bottom: A UML object model depicting ordered links

Moreover, in a UML object model, a link of an ordered property is labeled with an index, and requires that the index is unique and in order. In OWL2 there is no specific axiom for the representation of a UML ordered property link (7) or any link with a label. Due to this fact, we translate a UML ordered property link into OWL2 in four steps. First, declare an *index* property for each ordered property link that exist in an object model. The *index* property is declared in OWL2 as:

```
Declaration( ObjectProperty( index_P_# ) )
```

To make every ordered property link reachable while parsing a translated object model in OWL2, the name of an *index* property comprises three parts. First, *index* refers that the link of this objectproperty will represent an ordered property link. Second, *P* refers the name of an ordered property in a class model. Third, # is representing an index or a label on a link of an ordered property. The data type of an index can be "xsd:integer" [20] or "xsd:string" [20]. For example, for each labeled link 1 and 2 of the ordered property P shown at the bottom of Figure 4, we will declare an *index* property in OWL2 as:

```
Declaration(ObjectProperty( index_P_1 ))
Declaration(ObjectProperty( index_P_2 ))
```

Second, a domain and a range of each *index* property in OWL2 is same as the domain and the range of an ordered property. Therefore, each index property in OWL2 will be a subproperty of an ordered property P:

```
SubObjectPropertyOf( index_P_# P )
```

Third, all links of an *index* property must have an identical source and target (8). In OWL2 all links having an identical source and target are considered as one link. Therefore, we have also made the *index* property:

1. FunctionalObjectProperty so that one link of an *index* property may not lead to two individuals.

   ```
   FunctionalObjectProperty( index_P_# )
   ```

2. InverseFunctionalObjectProperty so that two links of an *index* property may not lead to one individual.

   ```
   InverseFunctionalObjectProperty( index_P_# )
   ```

Last, each *index* property in OWL2 is then instantiated among the respective individuals of a domain and a range class of an ordered property. For example the UML ordered property links $P(x1, y1, 1)$ and $P(x1, y2, 2)$ as shown at the bottom of Figure 4, is represented in OWL2 as:

```
ObjectPropertyAssertion( index_P_1 x1 y1)
ObjectPropertyAssertion( index_P_2 x1 y2)
```

6 Implementation of a Model Conformance Tool

6.1 Translator

We have implemented, the translations of UML class and object modeling concepts, as discussed in Section 4 and 5 into a translator, by using the model-to-text translation tool MOFScript [4]. The implemented translator in MOFScript allows us to automatically transform the UML class and object models into OWL2. The translator take a UML class model and an object model as an input in form of UML XMI 2.1 [15], which is parsed according to UML 2. The output of the translator is an ontology, which contain transformed object model and its class model in from of OWL2 functional syntax. The translator script can be downloaded from [3].

6.2 Reasoning Engine

The conformance of an object model against a class model is done by validating an output ontology using an OWL2 reasoner. The output ontology contains the transformed object model and its class model in form of OWL2. Since translation tool produces output ontology in OWL2 functional syntax, the validation of output ontology can be done by using any ontology reasoner which supports OWL2 functional syntax.

6.3 Conformance Report

A conformance report is an output of an OWL2 reasoner. After the validation of a transformed object model against its class model, a reasoner will produce the conformance report; this report will contain the details about the inconsistencies present in a transformed object model against its class model.

6.4 Example

We have translated the class model and its inconsistent object model as shown in Figure 1, into an OWL2 DL ontology, by using the implemented translator, the output ontology generated by the translator is as follows:

```
Declaration(Class(Operation))
SubClassOf( Operation_Direct Operation )
EquivalentClasses( Operation Operation_Direct )
DisjointClasses( Operation_Direct Parameter )
Declaration(Class(Parameter))
SubClassOf( Parameter_Direct Parameter )
EquivalentClasses( Parameter Parameter_Direct )
DisjointClasses( Parameter_Direct Operation )
Declaration(ObjectProperty( hasParameter ))
ObjectPropertyDomain(has_Parameter Operation)
ObjectPropertyRange(has_Parameter Parameter)
SubClassOf( ObjectOneOf( ATMLogin ) Operation_Direct )
SubClassOf( ObjectOneOf( ATMCardNumber ) Parameter_Direct )
SubClassOf( ObjectOneOf( ATMPinCode ) Parameter_Direct )
ObjectPropertyAssertion( hasParameter ATMLogin ATMCardNumber)
ObjectPropertyAssertion( hasParameter ATMLogin ATMPinCode)
SubObjectPropertyOf( index_hasParameter_1 hasParameter )
FunctionalObjectProperty( index_hasParameter_1 )
InverseFunctionalObjectProperty( index_hasParameter_1 )
ObjectPropertyAssertion( index_hasParameter_1 ATMLogin ATMCardNumber)
SubObjectPropertyOf( index_hasParameter_1 hasParameter )
FunctionalObjectProperty( index_hasParameter_1 )
InverseFunctionalObjectProperty( index_hasParameter_1 )
ObjectPropertyAssertion( index_hasParameter_1 ATMLogin ATMPinCode)
DifferentIndividuals( ATMLogin ATMCardNumber ATMPinCode )
```

The output of a translation tool which contains the transformed object model and its class model is then validated by a reasoner. The conformance report of the output ontology produced by a reasoner is as follows:

```
Consistent: No
Reason: Individual ATMLogin has more than one value
for the functional property index_hasParameter_1
```

The above conformance report is generated by using the Pellet OWL2 reasoner version 2.3.0 [2]. The conformance report clearly indicates the inconsistency that exists in the object model, that there exist more then one link of an ordered property *hasParameter* with an *index* label 1.

7 Conclusions

In this article we have presented an approach to reason about the conformance of a UML object model containing ordered properties against its class model using an OWL2 reasoner. Furthermore, we have discussed the implementation of the translations as an automatic model translation tool.

When comparing the scope of proposed translations with other approaches [19, 17, 12, 18, 22, 5, 1], we consider that we have made a contribution in the field of model conformance using Semantic Web technologies, by addressing the ordering of properties and the enforcement of the close-world restrictions in OWL2 DL.

The approach is fully automated thanks to the translation tool and the existing OWL2 reasoners. Unfortunately, the validation report generated by the reasoner is not always self-explanatory, because the relationship between UML concepts and OWL2 axioms is not so obvious. As a consequence, it is not always possible to immediately point out the cause of the problem based on these violations without a manual inspection of the validation report and the problematic object models. It would greatly add to the usefulness of the method to have some sort of automated discovery of the cause of violations.

As a future work, we would like to enhance the scope of the proposed translations by addressing the validation of other UML concepts, including composition, non-unique properties, data type and class enumeration.

References

1. Artale, A., Calvanese, D., Ibáñez García, A.: Full satisfiability of uml class diagrams. In: Proceedings of ER2010. pp. 317–331. ER'10, Springer-Verlag, Berlin, Heidelberg (2010)
2. Pellet: OWL 2 Reasoner for Java, available at `http://clarkparsia.com/pellet/`.
3. Model Validation MOFScript, available at `http://users.abo.fi/akhan/model_validation.m2t`.
4. MOFScript Homepage, available at `http://www.eclipse.org/gmt/mofscript/`.
5. Berardi, D., Calvanese, D., Giacomo, G.D.: Reasoning on uml class diagrams. Artif. Intell. 168(1-2), 70–118 (2005)
6. Chen, K., Sztipanovits, J., Neema, S.: Toward a semantic anchoring infrastructure for domain-specific modeling languages. In: Proceedings of EMSOFT 2005. pp. 35–43. EMSOFT '05, ACM, New York, NY, USA (2005)
7. Höglund, S., Khan, A.H., Liu, Y., Porres, I.: Representing and validating metamodels using owl 2 and swrl. In: Proceedings of the 9th Joint Conference on Knowledge-Based Software Engineering JCKBSE'10, Kaunas, Lithuania. JCKBSE'10 Conference Preceedings, JCKBSE (2010)
8. Kaneiwa, K., Satoh, K.: Consistency checking algorithms for restricted uml class diagrams. In: Proceedings of FoIKS2006, Springer. Springer (2006)
9. Kent, S.: Model Driven Engineering. In: Proc. of IFM International Formal Methods 2002. LNCS, vol. 2335. Springer-Verlag (2002)
10. Knorr, M., Alferes, J.J., Hitzler, P.: Local closed world reasoning with description logics under the well-founded semantics. Artificial Intelligence 175(9-10), 1528 – 1554 (2011)
11. Liu, Y., Höglund, S., Khan, A.H., Porres, I.: A feasibility study on the validation of domain specific languages using owl 2 reasoners. In: Proceedings of the 3rd Workshop on Transforming and Weaving Ontologies in Model Driven Engineering Malaga, Spain. CEUR Workshop Preceedings, CEUR (2010)

12. Machines, I.B., Group, O.M., Software, S.: Ontology definition metamodel (ODM), OMG Document ad/2003-02-23. Available at `http://www.omg.org/`.
13. Malgouyres, H., Motet, G.: A uml model consistency verification approach based on meta-modeling formalization. In: Proceedings of SAC2006. pp. 1804–1809. SAC '06, ACM, New York, NY, USA (2006)
14. OMG: UML 2.0 Infrastructure Specification (September 2003), document ptc/03-09-15, available at `http://www.omg.org/`.
15. OMG: XML Metadata Interchange (XMI) Specification, version 2.1 (September 2005), document formal/05-09-01, available at `http://www.omg.org/`.
16. OMG: UML 2.2 Superstructure Specification (February 2009), available at `http://www.omg.org/`.
17. Parreiras, F.S., Staab, S., Winter, A.: On marrying ontological and metamodeling technical spaces. In: ESEC-FSE '07: Proceedings. pp. 439–448. ACM, New York, NY, USA (2007)
18. Rahmani, Oberle, Dahms: An adjustable transformation from owl to ecore. In: MODELS 2010, Oslo, Norway, October 3-8, 2010. Proceedings. LNCS, Springer (2010)
19. Van Der Straeten, R.: Inconsistency Management in Model-driven Engineering. An Approach using Description Logics. Ph.D. thesis, Vrije Universiteit Brussel, Brussels, Belgium (September 2005)
20. Vlist, E.V.D. (ed.): RELAX NG: A Simpler Schema Language for XML. O'Reilly, California, CA, USA (2003)
21. W3: OWL 2 Document Overview (October 2009), available at `http://www.w3.org/TR/owl2-overview`.
22. Wang, S., Jin, L., Jin, C.: Ontology definition metamodel based consistency checking of uml models. In: CSCWD 2006. pp. 1–5 (may 2006)