Class and Object Model Conformance using OWL2 Reasoners

Ali Hanzala Khan Espen Suenson
Ivan Porres
Department 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 using the Web Ontology Language OWL2 and the Semantic Web Rule Language SWRL, 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 closed-world assumption when reasoning about UML models. Second, we want to preserve important model information such as composition and non-unique associations. Finally, model conformance is defined solely by OWL2 axioms where possible so that many cases can be reasoned about efficiently while SWRL is used to represent model composition constraints. We have also implemented an automatic model translation tool. The model translation tool takes as input an object model and its class model and produces an ontology that can be processed by an OWL2 reasoner to reveal the object model elements that do not conform to their class model.

Keywords: Model Validation, OWL2, Reasoning.

1 Introduction

Model Driven Engineering (MDE) [11] advocates the use of models to represent the most relevant design decisions in a software development process. A software development project involves the creation of many different models often using different modeling languages. 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) [5], 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. 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, first, if each object is a proper instance of a class and each link is an instance of an association depicted in the class model. Second, objects and links must preserve the uniqueness, multiplicity, source, target and composition constraints of their classes and associations. An example of a valid and an invalid object model and a class model is shown in Figure 1.

1.1 Overview of the Approach

In order to mechanically reason about model conformance we need to have a formal definition of the UML. In this article we have chosen the Web Ontology Language version 2 for Description Logic (OWL2 DL) [20] to formalize the UML class and object modeling concepts. There is 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, objects, 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 UML concepts such as composition

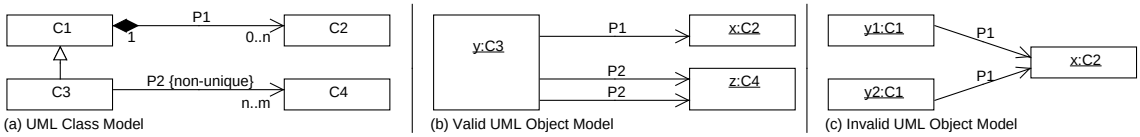

Figure 1: (a): A UML class model depicting a class hierarchy, a composition and a non-unique association. (b): A consistent UML object model conforms to the UML class model, and (c) An inconsistent object model due to the shared owner.

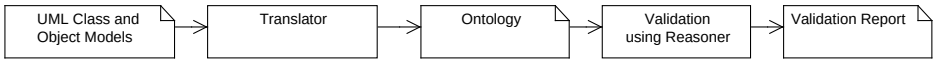

Figure 2: Automatic object and class model conformance process.

and non-unique association. Also, OWL2 uses open-world assumption, whereas, UML operates under the closed-world assumption [12], 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 composition, non-unique associations and the closed-world environment in OWL2 DL are the main contribution of the article and will be discussed in Sections 3 and 4 in detail.

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 [2]. The detail about the translation of a UML class and object modeling concepts into OWL2 will be discussed in Section 3, and the detail about the translation process will be discussed in Section 4.

Furthermore, the ontology generated by the translator, contains 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.

2 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 in Figure 1a. 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 in Figure 1b. In this section we give a formal definition of the UML class and object modeling concepts that we treat in our approach. The definition is given in terms of predicate logic. In this section we also motivate our choice of features that are included in the definition.

2.1 UML Classes and Objects

A UML class represents a set of objects that have the same characteristics [16]. A UML class C is defined as a unary predicate C in predicate logic.

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:

$$C(x) \tag{1}$$

Class Generalization. A class can be generalized by another class. In this case, a subclass inherits all traits and constraints of a more general superclass. Furthermore, in UML an object can only directly belong to one class and that any other classes it belongs to is through generalization. A generalization relationship where $C2$ is the subclass of $C1$, and x is representing any object of class $C2$, is formalized as:

$$\forall x.C2(x) \rightarrow C1(x) \tag{2}$$

Disjoint Classes. 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:

$$\forall x.\neg(C1(x) \wedge C2(x)) \tag{3}$$

2.2 UML Associations and Links

A UML binary association defines a relationship between two classes [16]. A UML link is an instance of an association. A link l of an association P connecting objects x and y is represented in predicate logic as:

$$P(x,y,l) \tag{4}$$

We often do not need to differentiate what link is used to connect two objects. Therefore is convenient for us to define the following:

$$\forall x,y,l.P(x,y,l) \rightarrow P(x,y) \tag{5}$$

Domain and Range. Every association connects a domain class and a range class. 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 association P is:

$$\forall x,y.P(x,y) \rightarrow C1(x) \wedge C2(y) \tag{6}$$

Multiplicity. The 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:

$$\forall x.C(x) \rightarrow n \leq \#\{y \mid P(x,y)\} \leq m \tag{7}$$

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

Unique and Non-unique Associations The UML associations can be labeled as unique or non-unique; a unique association restrict the objects of two classes to connect with each other more than one time. The restriction of unique association P is written in predicate logic as:

$$\forall x, y, l_1, l_2 . P(x, y, l_1) \wedge P(x, y, l_2) \rightarrow l_1 = l_2 \tag{8}$$

In the case of non-unique associations this restriction does not apply.

Bidirectional. Two UML associations may form a bidirectional and navigable relationship, provided that they have opposite domain and range classes. If P_1 and P_2 form an association then we require that:

$$\forall x, y . P_1(x, y) \leftrightarrow P_2(y, x) \tag{9}$$

Association Generalization. The association generalization allows the specialization of an existing association with new characteristics and new domain and range. Each instance of the specialized association is also an instance of the original association. If P_2 is a subassociation of P_1, this is expressed in predicate logic as:

$$\forall x, y . P_2(x, y) \rightarrow P_1(x, y) \tag{10}$$

If the subassociation specifies its own multiplicity restrictions, these are applied separately to the subassociation, in the same manner as to an ordinary property.

Composition. In composition relationship, an object of a class is made up of parts that are the objects of another class. A composition relationship is a directed association between two classes, which is meant to express a "part-of" or "ownership" relationship. If there is a composition relationship from class C1 to class C2 we say that C2 is owned by C1. The direction is marked by the familiar filled diamond on the association line; the diamond is placed on that end of the line that connects to the owning class.

We use a single predicate *owns* to keep track of the composition relationships. If C1 owns C2 via a composition association consisting of P_1 and P_2, and P_1 is the property from C1 to C2, then:

$$\forall x \in C1, y \in C2 . P_1(x, y) \rightarrow owns(x, y) \tag{11}$$

Composition relationships are defined in UML by two constraints, exclusive ownership and acyclic. Exclusive ownership means that an object can have only one owner:

$$\forall x, y, z . owns(x, z) \wedge owns(y, z) \rightarrow x = y \tag{12}$$

Acyclicity means that an object cannot transitively become an owner of itself. A situation where an object x owns y, y owns z and z owns x is disallowed. A necessary and sufficient condition for acyclicity of *owns* is that the transitive closure of the relation is irreflexive. We can define $owns^+$, the transitive closure of *owns*, in the following way:

$$\forall x, y . owns(x, y) \rightarrow owns^+(x, y) \tag{13}$$

$$\forall x, y, z . owns^+(x, y) \wedge owns^+(y, z) \rightarrow owns^+(x, z) \tag{14}$$

Irreflexivity of the transitive closure is then simply expressed as:

$$\forall x . \neg owns^+(x, x) \tag{15}$$

3 Translation of UML Class and Object Models to OWL2 DL

In this section we show the translation of UML Class modeling concepts discussed in Section 2 in to OWL2 DL.

3.1 UML Classes Hierarchy

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 ) )
```

Furthermore, if there is a generalization (2) between any two UML classes C1 and C2, such that, C1 is a subclass of C2, we represent the relationship in OWL2 as:

```
SubClassOf( C1 C2 )
```

Moreover, in UML class generalization an object of a subclass will also belong its superclass (2). According to UML semantics of class membership, whenever we assert that a model element is an instance of a class, we also assert that it is not an instance of its subclasses. But in OWL2 it is not like that, to overcome this semantic gap between OWL2 and UML, we translate UML class in OWL2 as a union of two disjoint concepts.

```
EquivalentClasses( C ObjectUnionOf( C_Direct C1..Cn ) )
DisjointClasses( C_Direct ObjectUnionOf( C1..Cn ) )
```

First concept *C_Direct*, represents the collection of all those objects which are the direct instances of a class. And, second concept represents all objects that belong to its subclasses $C1,..,Cn$.

Furthermore, 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 (3) among classes is represented as:

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

3.2 Objects

Every object exist in an object model must belong to a specific class in a class model. In OWL2 the UML object (1) is represented as a class assertion and called an *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)
```

3.3 UML Association

A UML association defines a relationship between two classes. A UML association (6) is represented in OWL2 as an object property. An association P from a class C1 to a class C2 is represented in OWL2 as:

```
Declaration( ObjectProperty( P ) )
ObjectPropertyDomain( P C1 )
ObjectPropertyRange( P C2 )
```

3.4 Links

A UML association assertion between the objects in an object model is called a *link* (4). A link in OWL2 is represented as a property assertion. 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 negative assertion. 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.

3.5 Associations Multiplicity

A UML association defines the multiplicity (7), in a non negative integer that 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 $C1$ and $C2$ has a multiplicity constraint of n..m is represented in OWL2 as:

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

3.6 Unique and Non-Unique Associations

A UML association multiplicity can be unique or non-unique. A unique association does not allow two objects to link with each other more than one time, as shown in Figure 3c. In OWL2 the UML unique association is treated as normal object property, and the translation of multiplicity constraint of unique association is done by using OWL2 axioms: ObjectMinCardinality and ObjectMaxCardinality, as discussed in Section 3.5. However, in case of a non-unique associ-

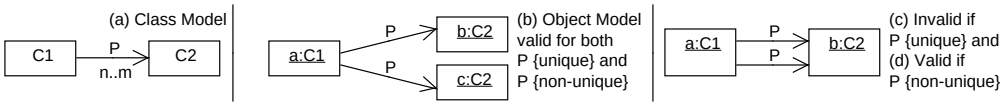

Figure 3: (a): A class model depicting an association P connection two classes, (b): A consistent object model for both unique and non-unique P, (c): An inconsistent object model if P is unique, and (d): A consistent object model if P is non-unique.

ation (8), there can be multiple links between the objects of a domain class and a range class, as shown in Figure 3d. OWL2 reasoner consider all links which have a common source and a target as one link, and to make reasoner to be able to consider all those links as different links, we have introduced an intermediate class in between a domain class and a range class of a non-unique association. As a consequence, every non-unique association P is translated in OWL2 as a combination of two object properties P_I and I_P. Where P_I connects a domain class to an intermediate class, and I_P connects an intermediate class to a range class of a non-unique association.

```
InverseFunctionalObjectProperty( P_I )
```

An inverse functional object property restricts an individual of a domain class to connect with more than one individuals of an intermediate class. Furthermore, we put a cardinality restriction of n..m on P_I by using OWL2 axioms: ObjectMinCardinality and ObjectMaxCardinality, as discussed in Section 3.5. Moreover, to ensures that the individuals of an intermediate class C_I connects one to one with the individuals of a range class, we have put the exact cardinality of one on the property I_P.

```
SubClassOf( C_I ObjectExactCardinality( 1 I_P ) )
```

Furthermore, a link of a non-unique association P from object a to b in an object model is then translated as the assertions of the object properties P_I and I_P as:

```
ObjectPropertyAssertion( P_I a C_I_# )
ObjectPropertyAssertion( I_P C_I_# b )
```

Where $C\_I\_\#$ is an individual of an intermediate class C_I, and # is an auto generated unique number which is responsible to create a distinction between the identical links of a non-unique association.

3.7 Bidirectionality

If P_1 and P_2 are UML associations and both associations shares opposite domain and range, then both associations will be considered as bidirectional (9) of each other. The UML bidirectionality between the associations $P1$ and $P2$ is express in OWL2 as:

```
InverseObjectProperty( P1 P2 )
```

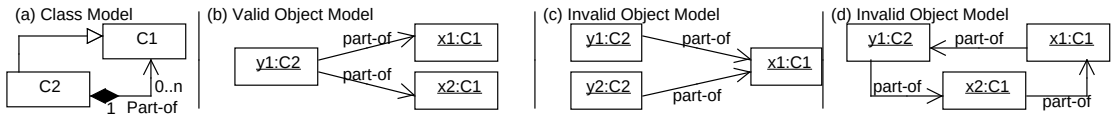

Figure 4: (a): Class model representing composition, (b): Consistent object model, (c): Inconsistent object model due to the shared owner, (d): Inconsistent object model due to the cyclic owner.

3.8 Association generalization

An association can be generalized by another association, the association generalization (10) is also known as association subsetting. The association subseting between association P_1 and P_2, where P_2 is a subassociation of P_1 is express in OWL2 as:

```
SubObjectPropertyOf( P2 P1 )
```

Similar to class generalization, the generalized association also inherits all the attributes of a parent association such as: domain and range. We can reassign a domain and a range of a subassociation, provided that the new domain and the range of a subassociation are the subclasses of the domain and the range of a parent association.

3.9 Composition

In composition (11), an object of a class is made up of parts that are the objects of another class. The composition has two constraints exclusive ownership (12) and acyclic (15). Exclusive ownership, an object can only become a part of one single object, or an object may only have one single owner as shown in Figure 4c. Furthermore, an acyclic means, an object cannot become a part of itself or an object cannot transitively become an owner of itself as shown in Figure 4d. To translate composition shown in Figure 4a into OWL2, we first define an object property named "part-of".

```
Declaration( ObjectProperty( part-of ) )
```

Next, we consider the exclusive ownership constraint of composition on the owning end of a composite relationship. To implement the single owner requirement of a composition relationship in OWL2, we have firstly, defined the global property *owns* as:

```
InverseFunctionalObjectProperty( owns )
```

The inverse functional property will restrict the individuals of containing class to link with more than one individuals of owning class. Secondly, we make the composite relationship "part-of", a subproperty of the global property *owns*:

```
SubObjectPropertyOf( part-of owns )
```

Moreover, to capture the acyclic requirement of composition, we define another global property contains which is transitive and irreflexive at same time. Transitivity will capture the self ownership and irreflexive will not allow the individual to become an owner of itself. This is equivalent to saying that the transitive closure of the ownership property is irreflexive:

```
IrreflexiveObjectProperty( contains )
```

However, it is not possible in OWL2 DL, to combine a cardinality restriction with transitive properties [9]. If we could do so, the logic system would no longer be decidable, and we would not be able to use a fully automatic reasoner to carry out validation. To solve this problem we have expressed the irreflexivity in OWL2 and transitivity in SWRL:

$$contains(?x, ?y) \wedge contains(?y, ?z) \implies contains(?x, ?z)$$

In above SWRL rule, contains represents the aggregation of model elements. Consequently, each object that owns another object must also contain it. We express this by defining *owns* a subset of *contains*:

```
SubObjectPropertyOf( owns contains )
```

4 Implementation of a Model Conformance Tool

4.1 Translator

We have implemented the translations of UML class and object modeling concepts, as discussed in Section 3, into a translator tool by using the model-to-text transformation tool MOFScript [2]. The implemented translator allows us to automatically transform the UML class and object models into OWL2. The translator takes a UML 2 class model and an object model as input in the form of UML XMI 2.1 [15]. The output of the translator is an ontology, which contain transformed object model and its class model in from of OWL2 functional syntax OWL2fs and SWRL. The translator script can be downloaded from [1].

As an example, we have translated the UML class model as shown in Figure 3a, and the object model depicting multiple links of unique association shown in Figure 3c into OWL2 DL ontology using our translator. The output ontology generated by the translator is as follows:

```
// UML Class Model into OWL2
Declaration(Class(C1))
SubClassOf( C1_Direct C1 )
EquivalentClasses( C1 C1_Direct )
DisjointClasses( C1_Direct C2 )
Declaration(Class(C2))
SubClassOf( C2_Direct C2 )
EquivalentClasses( C2 C2_Direct )
DisjointClasses( C2_Direct C1 )
Declaration(ObjectProperty( P ))
ObjectPropertyDomain(P C1)
ObjectPropertyRange(P C2)
InverseFunctionalObjectProperty( P_I )
ObjectPropertyDomain(P_I C1)
ObjectPropertyRange(P_I C_I)
SubClassOf(C1 ObjectMinCardinality(1 P))
SubClassOf(C1 ObjectMaxCardinality(5 P))
Declaration(ObjectProperty( I_P ))
ObjectPropertyDomain(I_P C_I)
ObjectPropertyRange(I_P C2)
SubClassOf(C1 ObjectExactCardinality(1 I_P))
```

```
// UML Object Model into OWL2
SubClassOf( ObjectOneOf( a ) C1_Direct )
SubClassOf( ObjectOneOf( b ) C2_Direct )
ObjectPropertyAssertion( P a b)
ObjectPropertyAssertion( P_I a C_I_1 )
ObjectPropertyAssertion( I_P C_I_1 b )
ObjectPropertyAssertion( P a b)
ObjectPropertyAssertion( P_I a C_I_1 )
ObjectPropertyAssertion( I_P C_I_1 b )
EquivalentClasses( C_I ObjectOneOf( C_I_1 ))
DifferentIndividuals( a C_I_1 b  )
```

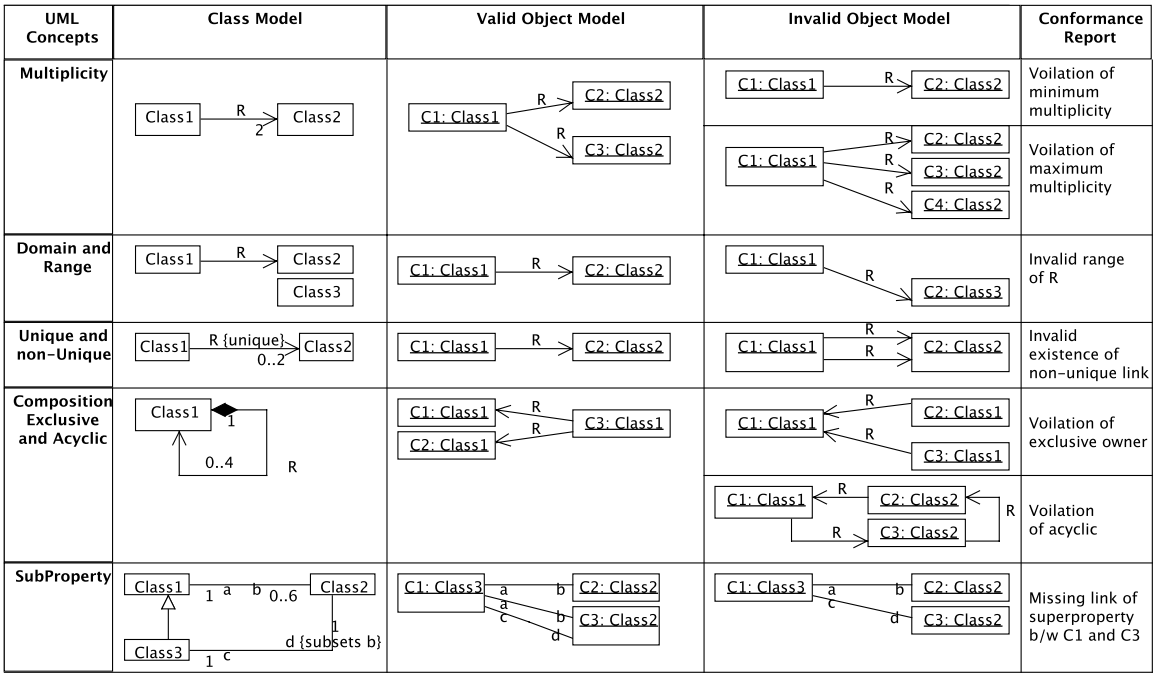

Figure 5: List of test cases and the conformance report summary of invalid object models.

4.2 Reasoning and Conformance Report

The conformance of an object model against a class model is done by validating the output ontology generated by the translation tool using an OWL2 reasoner. This step is automatic. Once completed, the OWL2 reasoner produces a report as its output. This report contains details about possible inconsistencies present in the ontology. The conformance report of an output ontology shown in Section 4.1 is as follows:

```
Consistent: No
Reason: Individual a is sameAs and differentFrom C_I_1 at the same time
```

This means that the individual *a* cannot really belong to class *C_1*. A conformance report generated by an OWL2 reasoner is not always self-explanatory. The analysis of the ontology and the conformance report reveled that, the individuals *a* and *C_I_1* are connected more than one time with the links of an inverse functional object property *I_P*. Since an inverse functional object property does not allow multiple links to connect with one individual, the reasoner complains that the output ontology is inconsistent.

4.3 Tool Validation

The proposed model translation tool has been validated by a suite of test cases, that covers all class and object modeling concepts discussed in Section 3. Each test case includes a class model, a valid object model, and an invalid object model. Each test case is transformed into OWL2 using the proposed translation tool and validated by using an OWL2 reasoner. The detail of test cases and the summary of conformance report is expressed in Figure 5.

4.4 Complexity

We have used OWL2 DL for the representation of UML concepts where ever it's possible, and used a decidable fragment of SWRL only for the representation of a model composition constraint. OWL2 DL is the standardized formalism of DL which is equivalent to $\mathscr{SHOIN}(\mathscr{D}^+)$. The complexity of OWL2 DL with regard to the reasoning problems of ontology consistency and instance checking is NEXPTIME complete [8], whereas the complexity of SWRL is undecidable [7, 22]. The combination of OWL2 DL and SWRL becomes undecidable [7]. However, if all atoms that exist in the SWRL rule use OWL2 class and property names are restricted to known individuals, then the SWRL rule is considered as DL-Safe rule and becomes decidable [22, 6]. The data complexity of query answering in the decidable fragment of SWRL is deterministic exponential time [22].

The ontology generated by the translation tool is written by using both OWL2 DL and the decidable fragment of SWRL. In the light of above discussion, we can conclude that the ontology generated by the translation tool is decidable, and the complexity with regard to the reasoning problems such as ontology consistency and instance checking is deterministic NEXPTIME complete.

5 Conclusions and Related Work

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

The approach is fully automated thanks to the translation tool and the existing OWL2 reasoners. Since the translation tools accept standard UML models serialized using the XMI standard, the approach can be easily integrated with existing UML modeling tools. Unfortunately, the validation report generated by OWL2 reasoners is not always self-explanatory, because the relationship between UML concepts and OWL2 axioms is not always obvious. As a consequence, it is not always possible to immediately point out the cause of the problem based on these violations without 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.

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, 13, 18, 21, 4, 3]. However, to our knowledge, none of them has addressed the reasoning of composition and non-unique properties in detail, neither the enforcement of the closed-world restrictions in OWL2 DL. These works focus on the problem of class model satisfiability, e.g. a class model can generate consistent object models or not. Furthermore, the validation of UML models using OCL has been discussed by several authors, including [10, 14]. In OCL, the model conformance rules must be defined explicitly based on the syntax of the UML models. In our approach, model conformance is defined on the semantic interpretation of the models. The difference is that while OCL must define a large number of well-formed rules for different variations and combinations of model elements, a logic approach requires a smaller number of axioms that are often simpler.

As a future work, we would like to enhance the scope of the proposed translations, by addressing the validation of other UML concepts for example: interfaces, data type, class enumeration and class visibility.

References

[1] Model Validation MOFScript, available at http://users.abo.fi/akhan/model_validation.m2t.

[2] MOFScript Homepage, available at http://www.eclipse.org/gmt/mofscript/.

[3] 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)

[4] Berardi, D., Calvanese, D., Giacomo, G.D.: Reasoning on uml class diagrams. Artif. Intell. 168(1-2), 70–118 (2005)

[5] 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)

[6] Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)

[7] Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A semantic web rule language combining OWL and RuleML (2004), availible at http://www.w3.org/Submission/SWRL/

[8] Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From $\mathscr{SHIQ}$ and RDF to OWL: The making of a web ontology language. J. of Web Semantics 1(1), 7–26 (2003), download/2003/HoPH03a.pdf

[9] Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In: Proceedings of LPAR 1999. pp. 161–180. Springer-Verlag, London, UK (1999)

[10] Kaneiwa, K., Satoh, K.: Consistency checking algorithms for restricted uml class diagrams. In: Proceedings of FoIKS2006, Springer. Springer (2006)

[11] Kent, S.: Model Driven Engineering. In: Proc. of IFM International Formal Methods 2002. LNCS, vol. 2335. Springer-Verlag (2002)

[12] 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)

[13] Machines, I.B., Group, O.M., Software, S.: Ontology definition metamodel (ODM), OMG Document ad/2003-02-23. Available at http://www.omg.org/.

[14] 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)

[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] W3: OWL 2 Document Overview (October 2009), available at http://www.w3.org/TR/owl2-overview.

[21] Wang, S., Jin, L., Jin, C.: Ontology definition metamodel based consistency checking of uml models. In: CSCWD 2006. pp. 1–5 (may 2006)

[22] Zhao, Y., Pan, J.Z., Jekjantuk, N., Henriksson, J., Groner, G., Ren, Y.: Most project - definition of language hierarchy (2008), availible at https://www.most-project.eu/admin/xinha/plugins/ExtendedFileManager/images/Deliverables/MOST_Deliverable_D3.1.pdf