Reasoning on UML Class Diagrams is EXPTIME-hard

Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
lastname@dis.uniroma1.it

Abstract

UML is the de-facto standard formalism for software design and analysis. One of the most important components of UML are *class diagrams*, which model the information on the domain of interest in terms of objects organized in classes and relationships between them. To support the design of large-scale industrial applications, CASE tools should be equipped with automated reasoning capabilities in order to detect relevant formal properties of UML diagrams, such as inconsistencies or redundancies. However, reasoning on UML class diagrams is fairly complex: in this paper we show that it is EXPTIME-hard.

1 Introduction

UML (Unified Modeling Language) [25] is the de-facto standard formalism for the analysis and design of software. One of the most important components of UML are *class diagrams*, which model the information on the domain of interest in terms of objects organized in classes and relationships between them[1].

Several works propose to describe UML class diagrams using various kinds of formal systems [18, 17, 22, 10]. Using such formal systems, one can potentially reason on UML class diagrams, and formally prove properties of interest through inference. In order to select the appropriate kind of formal tool for UML class diagrams, a fundamental question needs to be addressed: What is the computational complexity of reasoning on UML class diagrams? That is, independently of the particular formal tool adopted for describing such diagrams, how difficult is it to reason about them from the computational point of view?

In this paper we address this question resorting to results developed through the years in Description Logics (DLs) [1]. It is well known that DLs are logics that admit decidable reasoning and that are specifically designed for the conceptual representation of an application domain in terms of classes and relationships between classes. Representing conceptual data models by means of DLs has gathered consensus over the years, cf. [6, 7, 15, 24, 26, 13, 16, 12, 20, 21, 14, 23].

Our contribution is to show that reasoning on UML class diagrams is EXPTIME-hard even under fairly restrictive assumptions, namely: only binary associations, only

[1]In this paper we deal with UML class diagrams for the *conceptual perspective*, as opposed to the *implementation perspective*, see, e.g., [19].

minimal multiplicity constraints, generalizations with disjointness and covering constraints. We get this result by exhibiting a polynomial reduction from reasoning in the basic DL $\mathcal{ALC}$[2] [1], which is EXPTIME-complete. In particular, we show that every $\mathcal{ALC}$ knowledge base can be expressed as a UML class diagram preserving soundness and completeness of reasoning. This possibility is quite surprising, since UML class diagrams apparently have very limited means to express negative and disjunctive information, namely disjointness and covering constraints in generalization hierarchies. Note that we do not consider arbitrary OCL [25] constraints, which in their full generality have the same power of full first-order logic and hence lead to undecidability. Instead, $\mathcal{ALC}$ is equipped with unrestricted negation and disjunction. That is, it is able to treat negative information in the same way as positive one, and to reason by cases to fully take into account disjunctive information.

The rest of the paper is organized as follows. In Section 2 we give some preliminary notions on DLs that we use later on. In Section 3 we present our EXPTIME-hardness result for reasoning on UML class diagrams. Finally, in Section 4, we draw some conclusions. Throughout the paper we assume that the reader is familiar with UML class diagrams [25].

2 The Description Logics $\mathcal{ALC}$ and $\mathcal{ALC}^-$

$\mathcal{ALC}$ is a standard DL, that represents knowledge in terms of concepts (classes) and roles (binary relations). Let A and P denote atomic concepts and atomic roles respectively. An $\mathcal{ALC}$ concept C is built according to the following syntax:

$$C \ ::= \ A \ \mid \ \neg C \ \mid \ C_1 \sqcap C_2 \ \mid \ \exists P.C$$

We also introduce the standard abbreviations: $C_1 \sqcup C_2$ for $\neg(\neg C_1 \sqcap \neg C_2)$ and $\forall P.C$ for $\neg\exists P.\neg C$. An $\mathcal{ALC}$ Knowledge Base (KB) is constituted by a finite set of *inclusion assertions* of the form $C_1 \sqsubseteq C_2$, with C_1 and C_2 arbitrary concept expressions. We also consider *primitive* inclusions assertions, i.e., assertions of the form $A \sqsubseteq C$, where A is an atomic concept and C is an arbitrary concept. A *primitive KB* is constituted by a finite set of primitive inclusion assertions.

As usual in DLs, the semantics of $\mathcal{ALC}$ is specified through the notion of interpretation. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of an $\mathcal{ALC}$ KB $\mathcal{K}$ is constituted by an *interpretation domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns to each concept C a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and to each role P a subset $P^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$), such that the following conditions are satisfied:

$$\begin{array}{rclrcl}
A^{\mathcal{I}} & \subseteq & \Delta^{\mathcal{I}} & (C_1 \sqcap C_2)^{\mathcal{I}} & = & C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} & (\exists P.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid \exists b.\,(a,b) \in P^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}
\end{array}$$

An interpretation $\mathcal{I}$ *satisfies* an inclusion assertion $C_1 \sqsubseteq C_2$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$. An interpretation that satisfies all assertions in a KB $\mathcal{K}$ is called a *model* of $\mathcal{K}$. A KB $\mathcal{K}$ is *satisfiable* if there exists a model of $\mathcal{K}$. A concept C is *satisfiable* in a KB $\mathcal{K}$ if there is a model $\mathcal{I}$ of $\mathcal{K}$ such that $C^{\mathcal{I}}$ is non-empty. An assertion α is *logically implied* by

[2]In this paper when we mention reasoning in a DL, we always intend reasoning over a knowledge base expressed in that DL.

$\mathcal{K}$ if all models of $\mathcal{K}$ satisfy α. It can be shown that all these reasoning tasks, namely KB satisfiability, concept satisfiability in a KB, and logical implication, are mutually reducible (in polynomial time). In spite of its simplicity, reasoning in $\mathcal{ALC}$ KBs is EXPTIME-complete [1].

The DL $\mathcal{ALC}^-$ is obtained from $\mathcal{ALC}$ by dropping intersection, and restricting the assertions to be primitive. Therefore, an $\mathcal{ALC}^-$ concept C is built as follows:

$$C \ ::= \ A \ \mid \ \neg A \ \mid \ A_1 \sqcup A_2 \ \mid \ \exists P.A \ \mid \ \forall P.A$$

where A denotes an atomic concept and P denotes an atomic role. An $\mathcal{ALC}^-$ KB is a finite set of primitive $\mathcal{ALC}^-$ inclusion assertions, i.e., inclusion assertions of the form $A \sqsubseteq C$ where C is an $\mathcal{ALC}^-$ concept. The semantics of $\mathcal{ALC}^-$ constructs and KBs is that of $\mathcal{ALC}$. For $\mathcal{ALC}^-$, we can define KB satisfiability, concept satisfiability in a KB, and logical implication, as for $\mathcal{ALC}$.

3 Hardness of reasoning on UML class diagrams

The design quality of UML class diagrams can be improved by checking relevant properties, such as, for example, consistency of the whole class diagram, class consistency, class subsumption, class equivalence [7, 15, 8]. From the formal point of view, the reasoning tasks necessary for checking these properties are mutually reducible to each other. Hence in the following, without loss of generality, we focus on class consistency only. Specifically, we show that class consistency in UML class diagrams with only disjointness and covering constraints is EXPTIME-hard. We prove the claim by a reduction from concept satisfiability in $\mathcal{ALC}$ KBs, which is EXPTIME-hard [1]. For lack of space we report only proof sketches. The full proofs may be found in [4]. We proceed in two steps:

1. First, we show that we can restrict the attention to atomic concept satisfiability in $\mathcal{ALC}^-$ KBs.

2. Then, we describe a reduction from atomic concept satisfiability in $\mathcal{ALC}^-$ KBs to class consistency in UML class diagrams.

For point (1), we resort to known results [9, 11].

Lemma 3.1 ([9]) *Concept satisfiability in an $\mathcal{ALC}$ KB can be linearly reduced to atomic concept satisfiability in a primitive $\mathcal{ALC}$ KB.*

Proof (sketch). Let $\mathcal{K}$ be an $\mathcal{ALC}$ KB and C an $\mathcal{ALC}$ concept. It is easy to see that C is satisfiable in $\mathcal{K}$ if and only if A_C is satisfiable in the KB consisting of the following two assertion:

$$\begin{array}{lcl} A_T & \sqsubseteq & \bigsqcap_{C_1 \sqsubseteq C_2 \in \mathcal{K}} (\neg C_1 \sqcup C_2) \sqcap \bigsqcap_{1 \leq i \leq n} \forall P_i.A_T \\ A_C & \sqsubseteq & A_T \sqcap C \end{array}$$

where A_C and A_T are new atomic concepts and $P_1, \ldots, P_n$ are all atomic roles appearing in $\mathcal{K}$ and C. $\square$

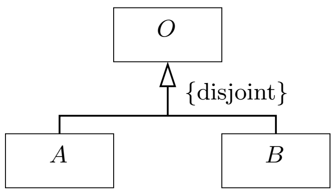

Figure 1: UML encoding of the assertion $A \sqsubseteq \neg B$

Below we assume, without loss of generality, that primitive $\mathcal{ALC}$ KBs are in negation normal form. Indeed, every primitive $\mathcal{ALC}$ KB can be put in negation normal form in linear time.

Given a primitive $\mathcal{ALC}$ KB $\mathcal{K}$ (in negation normal form), we construct a primitive $\mathcal{ALC}^-$ KB $\mathcal{K}'$ by recursively replacing each $\mathcal{ALC}$ assertion in K that is not already a (primitive) $\mathcal{ALC}^-$ assertion as follows:

- $A \sqsubseteq C_1 \sqcap C_2$ is replaced by $A \sqsubseteq C_1$ and $A \sqsubseteq C_2$;
- $A \sqsubseteq C_1 \sqcup C_2$ is replaced by $A \sqsubseteq A_1 \sqcup A_2$, $A_1 \sqsubseteq C_1$ and $A_2 \sqsubseteq C_2$, where A_1 and A_2 are new atomic concepts;
- $A \sqsubseteq \forall P.C$ is replaced by $A \sqsubseteq \forall P.A_1$ and $A_1 \sqsubseteq C$, where A_1 is a new atomic concept;
- $A \sqsubseteq \exists P.C$ is replaced by $A \sqsubseteq \exists P.A_1$ and $A_1 \sqsubseteq C$, where A_1 is a new atomic concept.

Notice that the number of such replacements is linear, since for each occurrence of an $\mathcal{ALC}$ construct in $\mathcal{K}$ at most one replacement is done. The following result holds.

Lemma 3.2 ([11]) *An atomic concept A_0 is satisfiable in a primitive $\mathcal{ALC}$ KB $\mathcal{K}$ if and only if A_0 is satisfiable in the (primitive) $\mathcal{ALC}^-$ KB $\mathcal{K}'$ obtained as above.*

Next, we reduce concept satisfiability in a primitive $\mathcal{ALC}^-$ KB $\mathcal{K}'$ to class consistency in an UML class diagram $\mathcal{D}$. For each atomic concept A in $\mathcal{K}'$, we introduce a class A in $\mathcal{D}$. Additionally, we add a class O that generalizes (possibly indirectly) all classes in $\mathcal{D}$. O is also used to specify disjointness among classes (see later). For each atomic role P, we introduce an association P, involving the class O twice. Intuitively, using O in such a way, we do not constrain in any way the classes to which the component instances of P may belong. More classes and associations, as well as generalizations between O and the new classes, are added below as needed.

The assertions in the $\mathcal{ALC}^-$ KB $\mathcal{K}'$ are encoded in the class diagram $\mathcal{D}$ as follows:

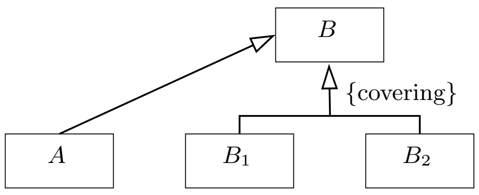

Figure 2: UML encoding of the assertion $A \sqsubseteq B_1 \sqcup B_2$

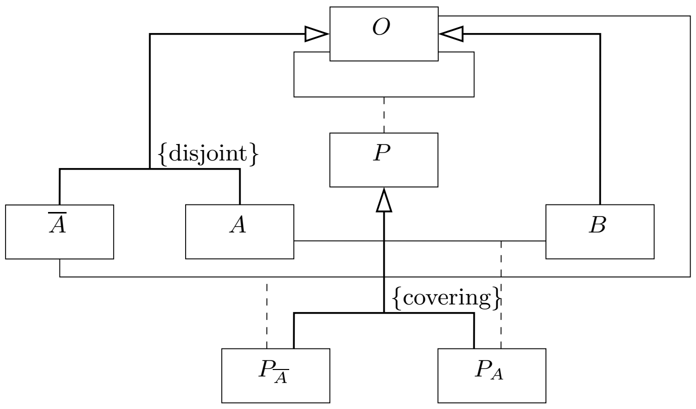

Figure 3: UML encoding of the assertion $A \sqsubseteq \forall P.B$

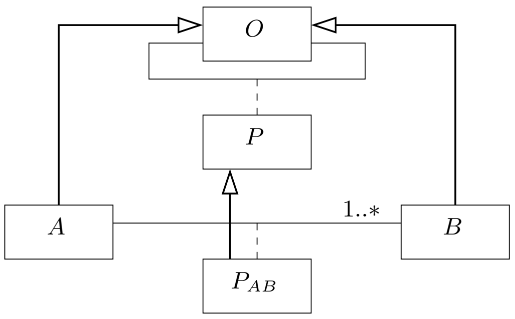

Figure 4: UML encoding of the assertion $A \sqsubseteq \exists P.B$

- For each assertion of the form $A \sqsubseteq B$, we introduce a generalization between the classes A and B (where A is the subclass).
- For each assertion of the form $A \sqsubseteq \neg B$, we construct the hierarchy in Figure 1, exploiting the superclass O to express disjointness between A and B.
- For each assertion of the form $A \sqsubseteq B_1 \sqcup B_2$, we introduce an auxiliary class B, and construct the hierarchy in Figure 2. Intuitively, being B a covering of B_1 and B_2, and A a subclass of B, it follows that A is a subclass of the union of B_1 and B_2.
- For each assertion of the form $A \sqsubseteq \forall P.B$, we introduce a new class $\overline{A}$ and two new binary associations P_A and $P_{\overline{A}}$ and we construct the portion of diagram in Figure 3, where A and $\overline{A}$ are disjoint and there is a generalization with covering constraint between P and its children P_A and $P_{\overline{A}}$. Note that A and B are the components of P_A, whereas $\overline{A}$ and O are the components of $P_{\overline{A}}$. Intuitively, the diagram enforces that each instance of A participating to P is in fact participating to P_A, and hence associated via P to an instance of B.
- For each assertion of the form $A \sqsubseteq \exists P.B$, we introduce a new binary association P_{AB} and we construct the portion of diagram shown in Figure 4. Note the proper

multiplicity constraint 1..$*$ on the participation of A to P_{AB}[3]. Intuitively, this implies that for each instance of A, there exists an instance of B related to it through P_{AB}, and hence through P.

Lemma 3.3 *Given a primitive $\mathcal{ALC}^-$ KB $\mathcal{K}'$, the size of the UML class diagram $\mathcal{D}$ constructed as above is linear in the size of $\mathcal{K}'$.*

Lemma 3.4 *An atomic concept A is satisfiable in an $\mathcal{ALC}^-$ KB $\mathcal{K}'$ if and only if the class A is consistent in the UML class diagram $\mathcal{D}$ constructed as above.*

Proof (sketch). We detail the proof only for $\mathcal{ALC}^-$ assertions of the form $A \sqsubseteq \forall P.B$ and $A \sqsubseteq \exists P.B$.

"$\Leftarrow$" Let $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ be an instantiation for $\mathcal{D}$. We show that $\mathcal{J}$ is also a model of all assertions in $\mathcal{K}'$.

- Each assertion of the form $A \sqsubseteq \forall P.B$ in $\mathcal{K}'$ corresponds, in $\mathcal{D}$, to the sub-diagram in Figure 3. $\mathcal{J}$ populates it according to the following constraints:

$$\begin{array}{rcl}
A^{\mathcal{J}} & \subseteq & O^{\mathcal{J}} \\
\overline{A}^{\mathcal{J}} & \subseteq & O^{\mathcal{J}} \\
A^{\mathcal{J}} \cap \overline{A}^{\mathcal{J}} & = & \emptyset \\
B^{\mathcal{J}} & \subseteq & O^{\mathcal{J}} \\
P^{\mathcal{J}} & \subseteq & O^{\mathcal{J}} \times O^{\mathcal{J}}
\end{array}
\qquad\qquad
\begin{array}{rcl}
P^{\mathcal{J}}_{\overline{A}} & \subseteq & \overline{A}^{\mathcal{J}} \times O^{\mathcal{J}} \\
P^{\mathcal{J}}_{A} & \subseteq & A^{\mathcal{J}} \times B^{\mathcal{J}} \\
P^{\mathcal{J}}_{A} & \subseteq & P^{\mathcal{J}} \\
P^{\mathcal{J}}_{\overline{A}} & \subseteq & P^{\mathcal{J}} \\
P^{\mathcal{J}} & \subseteq & P^{\mathcal{J}}_{A} \cup P^{\mathcal{J}}_{\overline{A}}
\end{array}$$

  From the constraints above, we get that $P^{\mathcal{J}}_{A} \cap P^{\mathcal{J}}_{\overline{A}} = \emptyset$. Therefore, if $x \in A^{\mathcal{J}}$ then for all $x' \in O^{\mathcal{J}}$ if $(x, x') \in P^{\mathcal{J}}$ then $(x, x') \in P^{\mathcal{J}}_{A}$ and therefore $x' \in B^{\mathcal{J}}$, i.e., $A^{\mathcal{J}} \subseteq \{x \in O^{\mathcal{J}} \mid \forall x' \in O^{\mathcal{J}}.\, (x, x') \in P^{\mathcal{J}} \supset x' \in B^{\mathcal{J}}\}$.

- Each assertion of the form $A \sqsubseteq \exists P.B$ in $\mathcal{K}'$ corresponds, in $\mathcal{D}$, to the sub-diagram shown in Figure 4. $\mathcal{J}$ populates it and satisfies the constraints $A^{\mathcal{J}} \subseteq O^{\mathcal{J}}$, $B^{\mathcal{J}} \subseteq O^{\mathcal{J}}$, $P^{\mathcal{J}} \subseteq O^{\mathcal{J}} \times O^{\mathcal{J}}$, $P^{\mathcal{J}}_{AB} \subseteq P^{\mathcal{J}}$, $P^{\mathcal{J}}_{AB} \subseteq A^{\mathcal{J}} \times B^{\mathcal{J}}$, and for each $x \in A^{\mathcal{J}}$ we have that $\sharp\{x' \in \Delta^{\mathcal{I}} \mid (x, x') \in P^{\mathcal{J}}_{AB}\} \geq 1$ (mandatory participation constraint). From these we get that for each $x \in A^{\mathcal{J}}$ there exists $x' \in O^{\mathcal{J}}$ such that $(x, x') \in P^{\mathcal{J}}$ and $x' \in B^{\mathcal{J}}$, i.e., $A^{\mathcal{J}} \subseteq \{x \in O^{\mathcal{J}} \mid \exists x' \in O^{\mathcal{J}}.\, (x, x') \in P^{\mathcal{J}} \wedge x' \in B^{\mathcal{J}}\}$.

"$\Rightarrow$" Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a model of $\mathcal{K}'$ with $A^{\mathcal{I}} \neq \emptyset$. We show that it can be seen as an instantiation of $\mathcal{D}$, once we assign a suitable extension to the auxiliary classes and roles introduced in the construction of $\mathcal{D}$. First, we define $O^{\mathcal{I}} = \Delta^{\mathcal{I}}$.

- For each assertion of the form $A \sqsubseteq \forall P.B$ in $\mathcal{K}'$, we have a fragment of $\mathcal{D}$ as in Figure 3. Let us define:
  - $\overline{A}^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
  - $P^{\mathcal{I}}_{A} = \{(x, x') \in P^{\mathcal{I}} \mid x \in A^{\mathcal{I}}\}$
  - $P^{\mathcal{I}}_{\overline{A}} = \{(x, x') \in P^{\mathcal{I}} \mid x \in \overline{A}^{\mathcal{I}}\}$

[3]In fact, in the case where we also have the assertion $A \sqsubseteq \forall P.B$ for some B, instead of proceeding as in Figure 4, we can simply add the cardinality constraint 1..$*$ to the association P_{AB} in Figure 3.

Then, by $A^{\mathcal{I}} \subseteq \{x \in \Delta^{\mathcal{I}} \mid \forall x' \in \Delta^{\mathcal{I}}.\, (x, x') \in P^{\mathcal{I}} \supset x' \in B^{\mathcal{I}}\}$, we get:

$$\begin{array}{rcl} A^{\mathcal{I}} & \subseteq & O^{\mathcal{I}} \\ \overline{A}^{\mathcal{I}} & \subseteq & O^{\mathcal{I}} \\ A^{\mathcal{I}} \cap \overline{A}^{\mathcal{I}} & = & \emptyset \\ B^{\mathcal{I}} & \subseteq & O^{\mathcal{I}} \\ P^{\mathcal{I}} & \subseteq & O^{\mathcal{I}} \times O^{\mathcal{I}} \end{array} \qquad \begin{array}{rcl} P_A^{\mathcal{I}} & \subseteq & A^{\mathcal{I}} \times B^{\mathcal{I}} \\ P^{\mathcal{I}} & \subseteq & P_A^{\mathcal{I}} \cup P_{\overline{A}}^{\mathcal{I}} \\ P_A^{\mathcal{I}} & \subseteq & P^{\mathcal{I}} \\ P_{\overline{A}}^{\mathcal{I}} & \subseteq & P^{\mathcal{I}} \end{array}$$

thus correctly capturing the fragment of $\mathcal{D}$.

- For each assertion of the form $A \sqsubseteq \exists P.B$ in $\mathcal{K}'$, we have a fragment of $\mathcal{D}$ as in Fig. 4. Let us define $P_{AB}^{\mathcal{I}} = \{(x, x') \in P^{\mathcal{I}} \mid x \in A^{\mathcal{I}}\}$. Then, by $A^{\mathcal{I}} \subseteq \{x \in \Delta^{\mathcal{I}} \mid \exists x' \in \Delta^{\mathcal{I}}.\, (x, x') \in P^{\mathcal{I}} \wedge x' \in B^{\mathcal{I}}\}$, we get that for each $x \in A^{\mathcal{I}}$ we have $\sharp\{x' \in \Delta^{\mathcal{I}} \mid (x, x') \in P_{AB}^{\mathcal{I}}\} \geq 1$, and we have that such an instantiation is correct for the fragment of $\mathcal{D}$. □

By Lemmata 3.1, 3.2, 3.3, 3.4, and EXPTIME-hardness of reasoning in $\mathcal{ALC}$ knowledge bases [1], we get our hardness result.

Theorem 3.5 *Class consistency in UML class diagrams is EXPTIME-hard.*

To show our hardness result, we have made no closure assumptions on the diagram. In UML class diagrams, closure assumptions of two forms are considered: (*i*) all classes not in the same hierarchy are a priori disjoint, and (*ii*) each object must be an instance of a *single* most specific class. Instead, in our framework we allow for two classes (possibly in a hierarchy) to have common instances, even when they do not have a common subclass.

It is easy to see that the results above continue to hold even if in UML class diagrams we make the assumption of mutual disjointness of classes that are not in a hierarchy. Indeed, it suffices to add to the diagram an auxiliary subclass for each pair of classes that correspond to concepts that are not trivially disjoint (here multiple inheritance is called in). In this way the default disjointness assumption will have no impact on the instantiation of the part of the diagram that is built as above.

4 Conclusions

We have shown that reasoning on UML class diagrams can be quite a complex task, since it is EXPTIME-hard. In [4] we show also that it can be reduced to reasoning in a DL KB, and thus is also in EXPTIME. Note that, in doing this, we have considered restricted forms of OCL constraints, which in their full generality have the same power of full first-order logic and hence lead to undecidability. This result suggests that it is highly desirable to provide automated reasoning support for detecting relevant properties of the diagram. The experimental results we reported in [3, 5, 4], while certainly limited and not providing a definitive answer, indicate that current state-of-the-art DL-based systems could serve as a core reasoning engine in advanced CASE tools, even if more research is needed in order to deal with complex UML class diagrams, and to perform reasoning tasks that involve reasoning on keys and identification constraints.

Finally, it is worth noting that the results presented here hold also for other conceptual modeling formalisms typically used in software engineering and databases.

In particular, the EXPTIME-completeness result applies to the Entity-Relationship model enhanced with ISA on entities and relationships [2].

References

[1] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press, 2003.

[2] Carlo Batini, Stefano Ceri, and Shamkant B. Navathe. *Conceptual Database Design, an Entity-Relationship Approach.* Benjamin and Cummings Publ. Co., Menlo Park, California, 1992.

[3] Daniela Berardi. Using description logics to reason on UML class diagrams. In *Proc. of the KI'2002 Workshop on Applications of Description Logics.* CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-63/`, 2002.

[4] Daniela Berardi, Andrea Calì, Diego Calvanese, and Giuseppe De Giacomo. Reasoning on UML class diagrams. Technical Report 11-03, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 2003.

[5] Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo. Reasoning on UML class diagrams using description logic based systems. In *Proc. of the KI'2001 Workshop on Applications of Description Logics.* CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-44/`, 2001.

[6] Sonia Bergamaschi and Claudio Sartori. On taxonomic reasoning in conceptual design. *ACM Trans. on Database Systems*, 17(3):385–422, 1992.

[7] Alexander Borgida. Description logics in data management. *IEEE Trans. on Knowledge and Data Engineering*, 7(5):671–682, 1995.

[8] Alexander Borgida and Ronald J. Brachman. Conceptual modeling with description logics. In Baader et al. [1], chapter 10, pages 349–372.

[9] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. *J. of Artificial Intelligence Research*, 1:109–138, 1993.

[10] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Reasoning on UML class diagrams in description logics. In *Proc. of IJCAR Workshop on Precise Modelling and Deduction for Object-oriented Software Development (PMD 2001)*, 2001.

[11] Diego Calvanese. *Unrestricted and Finite Model Reasoning in Class-Based Representation Formalisms.* PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1996. Available at `http://www.dis.uniroma1.it/pub/calvanes/thesis.ps.gz`.

[12] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 84–89, 1999.

[13] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Description logic framework for information integration. In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 2–13, 1998.

[14] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Data integration in data warehousing. *Int. J. of Cooperative Information Systems*, 10(3):237–271, 2001.

[15] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Description logics for conceptual data modeling. In Jan Chomicki and Günter Saake, editors, *Logics for Databases and Information Systems*, pages 229–264. Kluwer Academic Publisher, 1998.

[16] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Unifying class-based representation formalisms. *J. of Artificial Intelligence Research*, 11:199–240, 1999.

[17] Tony Clark and Andy S. Evans. Foundations of the Unified Modeling Language. In David Duke and Andy Evans, editors, *Proc. of the 2nd Northern Formal Methods Workshop*. Springer, 1997.

[18] Andy Evans, Robert France, Kevin Lano, and Bernhard Rumpe. Meta-modelling semantics of UML. In H. Kilov, editor, *Behavioural Specifications for Businesses and Systems*, chapter 2. Kluwer Academic Publisher, 1999.

[19] Martin Fowler and Kendall Scott. *UML Distilled – Applying the Standard Object Modeling Laguage*. Addison Wesley Publ. Co., Reading, Massachussetts, 1997.

[20] Enrico Franconi, Fabio Grandi, and Federica Mandreoli. A semantic approach for schema evolution and versioning in object-oriented databases. In *1st Int. Conf. on Computational Logic (CL 2000)*, volume 1861 of *Lecture Notes in Computer Science*, pages 1048–1062. Springer, 2000.

[21] Volker Haarslev and Ralf Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 161–168, 2001.

[22] David Harel and Bernhard Rumpe. Modeling languages: Syntax, semantics and all that stuff. Technical Report MCS00-16, The Weizmann Institute of Science, Rehovot, Israel, 2000.

[23] Carsten Lutz. Reasoning about entity relationship diagrams with complex attribute dependencies. In *Proc. of the 2002 Description Logic Workshop (DL 2002)*, pages 185–194. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-53/`, 2002.

[24] Deborah McGuinness and Jon R. Wright. Conceptual modelling for configuration: A description logic-based approach. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing J. – Special Issue on Configuration*, 12:333–344, 1998.

[25] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual.* Addison Wesley Publ. Co., Reading, Massachussetts, 1998.

[26] Ulrike Sattler. *Terminological Knowledge Representation Systems in a Process Engineering Application.* PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 1998.