3

Complexity of Reasoning

Francesco M. Donini

Abstract

We present lower bounds on the computational complexity of satisfiability and subsumption in several description logics. We interpret these lower bounds as coming from different "sources of complexity", which we isolate one by one. We consider both reasoning with simple concept expressions and with an underlying TBox. We discuss also complexity of instance check in simple ABoxes. We tried to enhance clarity and ease of presentation, sometimes sacrificing exhaustiveness for lack of space.

3.1 Introduction

Complexity of reasoning has been one of the major issues in the development of Description Logics (DL). This is because such logics are conceived [Brachman and Levesque, 1984] as the formal specification of subsystems for representing knowledge, to be used in larger knowledge-based systems. Since using knowledge means also to derive implicit facts from the told ones, the implementation of derivation procedures should take into account the optimality of reasoning algorithms. The study of optimal algorithms starts from the elicitation of the computational complexity of the problem the algorithm should solve. Initially, studies about the complexity of reasoning problems in DLs were more focused on polynomial-time versus intractable (NP- or coNP-hard) problems. The idea was that a Knowledge Representation system based on a DL with polynomial-time inference problems would guarantee timely answers to the rest of the system. However, once very expressive DLs with exponential-time reasoning problems were implemented [Horrocks, 1998b], it was recognized that knowledge bases of realistic size could be processed in reasonable time. This shifted most of the complexity analysis to DLs whose reasoning problems are ExpTime-hard, or worse.

This chapter presents some lower bounds on the complexity of basic reasoning tasks in simple DLs. The reasoning services taken into account are: first, satisfiability and subsumption of concept expressions alone (no TBox), then the same reasoning services considering a TBox also, and in the last part of the chapter, instance checking w.r.t. an ABox.

We show in detail some reductions from problems that are hard for complexity classes NP, coNP, PSpace, ExpTime, and from semidecidable problems to satisfiability/subsumption in various DLs. Then, we show how these reductions can be adapted to other DLs as well.

In several reductions, we use tableaux expansions to prove the correctness of the reduction. Thus, a secondary aim in this chapter is to show how tableaux are useful not only to devise reasoning algorithms and complexity upper bounds—as seen in Chapter 2—but also in finding complexity lower bounds. This is because tableaux untangle two different aspects of the computational complexity of reasoning in DLs:

- The first aspect is the structure of possible models of a concept. Such a structure is—in many DLs—a tree of individual names, linked by arcs labeled by roles. We consider such a tree an AND-tree, in the sense that all branches must be followed to obtain a candidate model. Following [Schmidt-Schauß and Smolka, 1991], we call *trace* each branch of such a tree. Readers familiar with tableaux terminology should observe that traces are not tableaux branches; in fact, they form a structure inside a single tableau branch.
- The second aspect is the structure of proofs or refutations. Clearly, if a trace contains an inconsistency—a *clash* in the terminology set up in Chapter 2, the candidate models containing this trace can be discarded. When all candidate models are discarded this way, we obtain a proof of subsumption, or unsatisfiability. Hence, the structure of refutations is often best viewed as an OR-tree of traces containing clashes.

Here we chose to mark the nodes with AND, OR, considering a satisfiability problem; if either unsatisfiability or subsumption are considered, AND-OR labels should be exchanged. Before starting with the various results, we elaborate more on this subject in the next paragraph.

3.1.1 Intuition: sources of complexity

The deterministic version of the calculus for $\mathcal{ALCN}$ in Chapter 2 can be seen as exploring an AND-OR tree, where an AND-branching corresponds to the (independent) check of all successors of an individual, while an OR-branching corresponds to the different choices of application of a nondeterministic rule.

Realizing that, one can see that the exponential-time behavior of the calculus is due to two independent origins: The AND-branching, responsible for the exponential size of a single candidate model, and the OR-branching, responsible for the exponential number of different candidate models. We call these two different combinatorial explosions *sources of complexity.*

3.1.1.1 OR-branching

The OR-branching is due to the presence of disjunctive constructors, which make a concept satisfiable by more than one model. The obvious disjunctive constructor is $\sqcup$, hence $\mathcal{ALU}$ is a good sublanguage to see this source of complexity. Recall that $\mathcal{ALU}$ allows one to form concepts using negation of concept names, conjunction $\sqcap$, disjunction $\sqcup$, universal role quantification $\forall R.C$, and unqualified existential role quantification $\exists$. This source of complexity is the same that makes propositional satisfiability NP-hard: in fact, satisfiability in $\mathcal{ALU}$ can be trivially proved NP-hard by rewriting propositional letters as atomic concepts, $\wedge$ as $\sqcap$, and $\vee$ as $\sqcup$. Many proofs of coNP-hardness of subsumption were found exploiting this source of complexity ([Levesque and Brachman, 1987; Nebel, 1988]), by reducing an NP-hard problem to non-subsumption. In Section 3.2.1, we show how disjunction can be introduced also by combining role restrictions and universal quantification, and in Section 3.2.2 by combining number restrictions and role intersection.

3.1.1.2 AND-branching

The AND-branching is more subtle. Its exponential behaviour is due to the interplay of qualified existential and universal quantifiers, hence $\mathcal{ALE}$ is now a minimal sublanguage of $\mathcal{ALCN}$ with these features. As mentioned in Chapter 2 one can see the effects of this source of complexity by expanding the tableau $\{D(x)\}$, when D is the following concept (whose pattern appears in many papers, from [Schmidt-Schauß and Smolka, 1991], to [Hemaspaandra, 1999])—see Chapter 2 for its general form:

$$\begin{aligned}
&\exists P_1.\forall P_2.\forall P_3.C_{11} \sqcap\\
&\exists P_1.\forall P_2.\forall P_3.C_{12} \sqcap\\
&\forall P_1.(\exists P_2.\forall P_3.C_{21} \sqcap\\
&\qquad \exists P_2.\forall P_3.C_{22} \sqcap\\
&\qquad \forall P_2.(\exists P_3.C_{31} \sqcap\\
&\qquad\qquad \exists P_3.C_{32}))
\end{aligned}$$

For each level l of nested quantifiers, we use a different role P_l (but using the same role R would produce the same results). The structure of the tableau for $\{D(x)\}$, which is the candidate model for D, is a binary tree of height 3: the nodes are the individual names, the arcs are given by the P_l-successor relation, and the branches are the traces in the tableau.

Each trace ends with an individual that belongs to C_{1i}, C_{2j}, C_{3k}, for $i, j, k \in \{1, 2\}$. Hence, a clash may be found independently in each trace, i.e., in each branch of the tree. To verify that this structure is indeed a model, one has to check every AND-branch of it; and branches can be exponentially many in the nesting of quantifiers.

This source of complexity causes an exponential number of possible *refutations* to be searched through (each refutation being a trace containing a clash).

This second source of complexity is not evident in propositional calculus, but a similar problem appears in predicate calculus—where the interplay of existential and universal quantifiers may lead to large models—and in Quantified Boolean Formulae.

Remark 3.1 For DLs that are not closed under negation, a source of complexity might appear in subsumption, while it may not in satisfiability. This is because C is subsumed by D iff $C \sqcap \neg D$ is unsatisfiable, where $\neg D$ may not belong to the same DL of C and D. ■

3.1.2 Overview of the chapter

We first present separately the effect of each source of complexity. In the next section, we discuss intractability results stemming from disjunction (OR-branching), which lead to coNP-hard lower bounds. We discuss both the case of plain logical disjunction (as the description logic $\mathcal{FL}$), and the case of disjunction arising from alternative identification of individuals ($\mathcal{ALEN}$). Then in Section 3.3 we present an NP lower bound stemming from AND-branching, namely a DL in which concepts have one candidate model of exponential size.

A PSpace lower bound combining the two sources of complexity is presented in Section 3.4, and then in Section 3.5 we show how axioms can combine in a succinct way the sources of complexity, leading to ExpTime-hardness of satisfiability.

In Section 3.6 we examine one of the first undecidability results found for a DL, using the powerful construct of role-value-maps—now recognized very expressive, because of this result.

Finally, we analyze intractability arising from reasoning with individuals in ABoxes (Section 3.7), and add a final discussion about the significance of these results—beyond the initial study of theoretical complexity of reasoning—also for benchmark testing of implemented procedures.

An appendix with a (hopefully complete) list of complexity results for satisfiability and subsumption closes the chapter.

Table 3.1. *Syntax and semantics of the description logic $\mathcal{FL}$. For $\mathcal{FL}^-$, omit role restriction.*

	concept expressions	semantics
concept name	A	$\subseteq \Delta^\mathcal{I}$
concept intersection	$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$
limited exist. quant.	$\exists R$	$\{x \in \Delta^\mathcal{I} \mid \exists y.\, (x,y) \in R^\mathcal{I}\}$
value restriction	$\forall R.C$	$\{x \in \Delta^\mathcal{I} \mid \forall y.\, (x,y) \in R^\mathcal{I} \rightarrow y \in C^\mathcal{I}\}$
	role expressions	**semantics**
role name	P	$\subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$
role restriction	$R\vert_C$	$\{(x,y) \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid (x,y) \in R^\mathcal{I} \wedge\ y \in C^\mathcal{I}\}$

3.2 OR-branching: finding a model

When the number of candidate models is exponential in the size of the concepts involved, a combinatorial problem is finding the right candidate model to check. In DLs, this may lead to NP-hardness of satisfiability, and coNP-hardness of subsumption.

3.2.1 Intractability in $\mathcal{FL}$

Brachman and Levesque [1984];[Levesque and Brachman, 1987] were the first to point out that a slight increase in the expressiveness of a DL may result in a drastic change in the complexity of reasoning. They called this effect a "computational cliff" of structured knowledge representation languages. They considered the language $\mathcal{FL}$, which admits concept conjunction, universal role quantification, unqualified existential quantification, and role restriction. For readability, the syntax and semantics of $\mathcal{FL}$ are recalled in Table 3.1.

Role restriction allows one to construct a subrole of a role R, i.e., a role whose extension is a subset of the extension of R. For example, the role $\mathsf{child}|_{\mathsf{male}}$ may be used for the "son-of" relation. Observe two properties of role restriction, whose proofs easily follow from the semantics in Table 3.1:

(i) for every role R, the role $R|_\top$ is equivalent to R;
(ii) for every role R, and concepts A, C, D, the concept $(\forall(R|_C).A) \sqcap (\forall(R|_D).A)$ is equivalent to $\forall(R|_{(C \sqcup D)}).A$.

The second property highlights that disjunction—although not explicitly present in the syntax of the language—arises from semantics.

Brachman and Levesque defined also the language $\mathcal{FL}^-$, derived from $\mathcal{FL}$ by omitting role restriction. They first showed that for $\mathcal{FL}^-$, subsumption can be decided by a structural algorithm, with polynomial time complexity, similar to the one shown in Chapter 2. Then they showed that subsumption in $\mathcal{FL}$ is coNP-hard, exhibiting the first "computational cliff" in description logics.

Since the original proof of coNP-hardness is somehow complex, we give here a simpler proof, found by Calvanese [1990]. The proof is based on the observation that if $C_1 \sqcup \cdots \sqcup C_n \equiv \top$, then, given a role R and a concept A, it is

$$\begin{aligned}(\forall(R|_{C_1}).A) \sqcap \cdots \sqcap (\forall(R|_{C_n}).A) &\equiv \text{(from (ii))} && (3.1)\\ \forall R|_{(C_1 \sqcup \cdots \sqcup C_n)}.A &\equiv && (3.2)\\ \forall R|_{\top}.A &\equiv \text{(from (i))} && (3.3)\\ \forall R.A & && (3.4)\end{aligned}$$

Moreover, observe that, for every role Q and every concept C, the disjunction $\exists Q \sqcup \forall Q.C$ is equivalent to the concept $\top$. Hence $\forall(R|_{\exists Q}).A \sqcap \forall(R|_{\forall Q.C}).A$ is equivalent to $\forall R.A$. These observations are the key to the reduction from tautology check of propositional 3DNF formulae to subsumption in $\mathcal{FL}$.

Theorem 3.2 *Subsumption in $\mathcal{FL}$ is coNP-hard.*

Proof Given an alphabet of propositional variables $L = \{p_1, \ldots, p_k\}$, define a propositional formula $F = G_1 \vee \cdots \vee G_n$ in 3DNF over L, where each disjunct G_i is made of three literals $l_i^1 \wedge l_i^2 \wedge l_i^3$, and for every $i \in \{1, \ldots, n\}$, and $j \in \{1, 2, 3\}$, each literal l_i^j is either a variable $p \in L$, or its negation $\overline{p}$.

Given a set of role names $\{R, P_1, \ldots, P_n\}$ (one role P_i for each variable p_i) and a concept name A, define the concept $C_F = (\forall R|_{C_1}.A) \sqcap \cdots \sqcap (\forall R|_{C_n}.A)$ where, for each $i \in \{1, \ldots, n\}$, C_i is the conjunction of three concepts $D_i^1 \sqcap D_i^2 \sqcap D_i^3$, and each D_i^j is

$$D_i^j = \begin{cases} \forall P_h.A, & \text{if } l_i^j = p_h \\ \exists P_h, & \text{if } l_i^j = \overline{p_h} \end{cases} \qquad \text{for } j \in \{1, 2, 3\},\ i \in \{1, \ldots, n\}$$

Then the claim follows from the following lemma. □

Lemma 3.3 *F is a tautology if and only if $C_F \equiv \forall R.A$.*

Proof The proof of the claim is straightforward; however, since it does not appear elsewhere but Calvanese's Master thesis (in italian), we present it here in full.

Only-if If F is a tautology, then $C_1 \sqcup \cdots \sqcup C_n \equiv \top$. This can be shown by contradiction: suppose $C_1 \sqcup \cdots \sqcup C_n$ is not equivalent to $\top$. Then, there exists an interpretation $\mathcal{I}$ in which there is an element $x \notin C_i^{\mathcal{I}}$, for every $i \in \{1, \ldots, n\}$. Since each $C_i = D_i^1 \sqcap D_i^2 \sqcap D_i^3$, it follows that for each i there is a $j \in \{1,2,3\}$ such that $x \notin D_i^j$. Define a truth assignment τ to L as follows. For each $h \in \{1,\dots,k\}$,

- $\tau(p_h) = \mathbf{false}$ iff $l_i^j = p_h$, and $x \notin D_i^j$
- $\tau(p_h) = \mathbf{true}$ iff $l_i^j = \overline{p_h}$, and $x \notin D_i^j$

Observe that it cannot be both $\tau(p_h) = \mathbf{false}$ and $\tau(p_h) = \mathbf{true}$ at the same time, since this would imply both $x \notin \exists P_h$, and $x \notin \forall P_h.A$, which is impossible since $\exists P_h \sqcup \forall P_h.A \equiv \top$. Evidently, τ assigns **false** to at least one literal for each disjunct of F, contradicting the hypothesis that F is a tautology. Therefore $C_1 \sqcup \cdots \sqcup C_n \equiv \top$.

The claim is now implied by equivalences (3.1)–(3.4).

If Suppose F is not a tautology. Then, there exists a truth assignment τ such that for each $i \in \{1,\dots,n\}$, there exists a $j \in \{1,2,3\}$ such that $\tau(l_i^j) = \mathbf{false}$.

Define an interpretation $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, with $\Delta^{\mathcal{I}}$ containing three elements x, y, z, such that $P_h^{\mathcal{I}} = (y,z)$ if $\tau(p_h) = \mathbf{false}$, and $P_h^{\mathcal{I}} = \emptyset$ otherwise. Moreover, let $A^{\mathcal{I}} = \emptyset$, and $R^{\mathcal{I}} = \{x,y\}$.

Observe that in this way, $y \in (\exists P_h)^{\mathcal{I}}$ iff $\tau(p_h) = \mathbf{false}$, and $y \in (\forall P_h.A)^{\mathcal{I}}$ iff $\tau(p_h) = \mathbf{true}$. This implies that $x \notin (\forall R.A)^{\mathcal{I}}$. To prove the claim, we now show that $x \in C_F^{\mathcal{I}}$.

Observe that, for each $i \in \{1,\dots,n\}$, there exists a $j \in \{1,2,3\}$ such that $\tau(l_i^j) = \mathbf{false}$. For such j, we show by case analysis that $y \notin (D_i^j)^{\mathcal{I}}$:

- if $l_i^j = p_h$ then $D_i^j = \forall P_h.A$, and in this case, $\tau(p_h) = \mathbf{false}$, hence $y \notin (\forall P_h.A)^{\mathcal{I}}$;
- if $l_i^j = \overline{p_h}$ then $D_i^j = \exists P_h$, and in this case, $\tau(p_h) = \mathbf{true}$, hence $y \notin (\exists P_h)^{\mathcal{I}}$.

Therefore, for every $i \in \{1,\dots,n\}$ it is $y \notin C_i^{\mathcal{I}}$. This implies that $(x,y) \notin R|^{\mathcal{I}}_{(C_1 \sqcup \cdots \sqcup C_n)}$, hence $x \in (\forall R|_{(C_1 \sqcup \cdots \sqcup C_n)}.A)^{\mathcal{I}}$, which is a concept equivalent to C_F. □

The above proof shows only that subsumption in $\mathcal{FL}$ is coNP-hard. However, role restrictions could be used also to obtain qualified existential quantification, since $\exists R.C = \exists R|_C$. Hence, $\mathcal{FL}$ contains also the AND-branching source of complexity. Combining the two sources of complexity, Donini *et al.* [1997a] proved a PSpace lower bound for subsumption in $\mathcal{FL}$, matching the upper bound found by Schmidt-Schauß and Smolka [1991].

3.2.2 Intractability in $\mathcal{FL}^-$ plus qualified existential quantification and number restrictions

As shown in Chapter 2, disjunction arises also from qualified existential quantification and number restrictions. This can be easily seen examining the construction of the tableau checking the satisfiability of the concept

$$(\exists R.A) \sqcap (\exists R.(\neg A \sqcap \neg B)) \sqcap (\exists R.B) \sqcap \leqslant 2\, R$$

in which, once three objects are introduced to satisfy the existentials, one has to choose between three non-equivalent identifications of pairs of objects, where only one identification leads to a consistent tableau.

Remark 3.4 When a DL includes number restrictions, also negation of concept names is included for free, at least from a computational viewpoint. In fact, a concept name A and its negation $\neg A$ can be coded as, say, $\geqslant 4\, R_A$ and $\leqslant 3\, R_A$ where R_A is a new role name introduced for A. Now these two concepts obey the same axioms of A and $\neg A$—namely, their conjunction is $\bot$ and their union is $\top$. Hence, everything we say about computational properties of DLs including $\mathcal{FL}^-$ plus number restrictions holds also for $\mathcal{AL}$ plus number restrictions. ■

We now present a proof of intractability based on this property. The reduction was first published by Nebel [1988], who reduced the NP-complete problem of SET SPLITTING [Garey and Johnson, 1979, p. 221], to non-subsumption in the DL of the BACK system, which included the basic $\mathcal{FL}^-$ plus intersection of roles, and number restrictions. SET SPLITTING is the following problem:

Definition 3.5 (set splitting) Given a collection $\mathcal{C}$ of subsets of a basic set S, decide if there exists a partition of S into two subsets S_1 and S_2 such that no subset of $\mathcal{C}$ is entirely contained in either S_1 or S_2. ■

We simplify the original reduction. We start from a variant of SET SPLITTING (still NP-complete) in which all $c \in \mathcal{C}$ have exactly three elements, and reduce it to satisfiability in $\mathcal{FL}^-$ plus qualified existential role quantification and number restrictions[1]. Since role intersection can simulate qualified existential role quantification (see next Section 3.2.2.1) this result implies the original one.

Theorem 3.6 *Satisfiability in $\mathcal{FL}^-\mathcal{EN}$ is* NP*-hard.*

Proof Let $S = \{1, \ldots, n\}$, and let $c_1, \ldots, c_k$ be the subsets of S. There exists a splitting of S iff the concept $D_1 \sqcap D_2 \sqcap D_3$ is satisfiable, where D_1, D_2, D_3 are defined as follows:

$$D_1 = \exists R.B_1 \sqcap \cdots \sqcap \exists R.B_n \quad (3.5)$$

$$D_2 = \forall R.(\leqslant 2\, Q_1 \sqcap \cdots \sqcap \leqslant 2\, Q_k) \quad (3.6)$$

$$D_3 = \leqslant 2\, R \quad (3.7)$$

[1] From Remark 3.4, this DL has the same computational properties of $\mathcal{ALEN}$ [Donini *et al.*, 1997a].

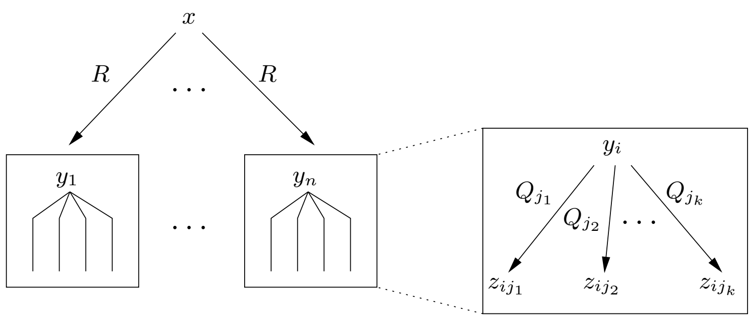


Fig. 3.1. The AND-tree structure of the tableau obtained by applying rules for $\sqcap$ and $\exists R.C$ to $D_1 \sqcap D_2 \sqcap D_3(x)$. Applying rule for $\leqslant 2\, R(x)$ would lead to several OR-branches (as many as the possible identifications of $y's$).

where each concept B_i codes which subsets element i appears in, as follows:

$$B_i = \sqcap_{j \mid i \in C_j} \exists Q_j.A_i$$

and concepts $A_1, \ldots, A_n$ are defined in such a way that they are pairwise disjoint—say, for $i \in \{1, \ldots, n\}$ let $A_i = \geqslant i\, R \sqcap \leqslant i\, R$. Intuitively, when tableaux rules dealing with $\sqcap$ and qualified existential quantification are applied to $D_1 \sqcap D_2 \sqcap D_3(x)$, one obtains a tableau whose tree structure of individual names can be visualized as in Figure 3.1. The rest of the proof strictly follows the original one [Nebel, 1988], hence we do not present it here. The intuition is that D_3 forces to identify all y's generated by D_1 into two successors of the root individual name x. Such identifications correspond to the sets S_1 and S_2. Then D_2 forces the split of each 3-subset, since it makes sure that neither of these successors has more than two Q_j-successors, and thus both have at least one Q_j-successor (since there are three of them). □

We clarify the construction and show its relevant properties on an example.

Example 3.7 Suppose $S = \{1, 2, 3, 4\}$, and let $c_1 = \{1, 2, 4\}, c_2 = \{2, 3, 4\}, c_3 = \{1, 3, 4\}$. Applying the tableau rules of Chapter 2 to D_1, one obtains the following tree of individual names (definitions of each B_i are expanded):

$$D_1(x)\begin{cases} R(x,y_1)\quad B_1(y_1) & \begin{cases} Q_1(y_1,z_{11}) & A_1(z_{11})\\ Q_3(y_1,z_{13}) & A_1(z_{13})\end{cases}\\ R(x,y_2)\quad B_1(y_1) & \begin{cases} Q_1(y_2,z_{21}) & A_2(z_{21})\\ Q_2(y_2,z_{22}) & A_2(z_{22})\end{cases}\\ R(x,y_3)\quad B_1(y_1) & \begin{cases} Q_2(y_3,z_{32}) & A_3(z_{32})\\ Q_3(y_3,z_{33}) & A_3(z_{33})\end{cases}\\ R(x,y_4)\quad B_1(y_1) & \begin{cases} Q_1(y_4,z_{41}) & A_4(z_{41})\\ Q_2(y_4,z_{42}) & A_4(z_{42})\\ Q_3(y_4,z_{43}) & A_4(z_{43})\end{cases}\end{cases}$$

where the individual names $y_1,\ldots,y_4$ stand for the four elements of S, and each z_{ij} codes the fact that element i appears in subset c_j. Because of assertions $A_i(z_{ij})$, no two z's disagreeing on the first index—e.g., z_{32} and z_{42}—can be safely identified, since they must satisfy assertions on incompatible A's. This is the same as if the constraints $z_{ij} \neq z_{hj}$, for all $i,h \in \{1,\ldots,|S|\}$ with $i \neq h$, and all $j \in \{1,\ldots,|\mathcal{C}|\}$ were present.

Now D_3 states that $y_1,\ldots,y_4$ must be identified into only two individual names. Observe that identifying y_2, y_3, y_4 leads to an individual name (say, y_2) having among others, three unidentifiable Q_2-fillers z_{22}, z_{32}, z_{42}. But D_2 states that all R-fillers of x, including y_2, have no more than 2 fillers for Q_2. This rules out the identification of y_2, y_3, y_4 in the tableau. Observe that this identification corresponds to a partition of S in $\{1\}$ and $\{2,3,4\}$ which is not a solution of SET SPLITTING because the subset c_2 is not split. Following the same line of reasoning, one could prove that the only identifications of all R-fillers into two individual names, leading to a satisfiable tableau, are one-one with solutions of SET SPLITTING. ■

The same reduction works for non-subsumption, since $D_1 \sqcap D_2 \sqcap D_3$ is satisfiable iff $D_1 \sqcap D_2$ is not subsumed by $\neg D_3 \equiv\, \geqslant 3\, R$. This type of reduction was also applied (see [Donini *et al.*, 1999]) to prove that subsumption in $\mathcal{ALNI}$ is coNP-hard, where $\mathcal{ALNI}$ is the DL including $\mathcal{AL}$, number restrictions and inverse roles.

Observe that also $\mathcal{FL}^{-}\mathcal{EN}$ contains the AND-branching source of complexity, since qualified existential restriction is present. With a more complex reduction from Quantified Boolean Formulae, combining the two sources of complexity, satisfiability and non-subsumption in $\mathcal{ALEN}$ has been proved PSPACE-complete by Hemaspaandra [1999].

Note that in the above proof of intractability, pairwise disjointness of $A_1,\ldots,A_n$ could be also expressed by conjoining $\log n$ concept names and their negations in all possible ways. Hence, the proof needs only the concept $\leqslant 2\, R$, and when qualified existentials are simulated by subroles, only $\geqslant 1\, R$ is used. This shows that the above proof of intractability is quite sharp: intractability raises independently of the size of the numbers involved. The computational cliff is evident if one moves to having 0 and 1 only in number restrictions, that leads to so-called *functional roles*—since the assertion $\leqslant 1\, R(x)$ forces R to be a partial function from x. In that case, the tractability of a DL can be usually established, e.g., the DL of the system Classic [Borgida and Patel-Schneider, 1994]. The intuitive reason for tractability of functional roles can be found in the corresponding tableau rules, which for number restrictions of the form $\leqslant 1\, R(x)$ become *deterministic*: there is no choice in identifying individuals names $y_1, \ldots, y_k$ which are all R-fillers for x, but to collapse them all into one individual.

3.2.2.1 Simulating $\exists R.C$ with role conjunction

Donini *et al.* [1997a] showed that a concept D containing qualified existential role quantifications $\exists R.C$ is satisfiable iff the concept $\widetilde{D}$ is satisfiable, where in $\widetilde{D}$ each occurrence of a concept $\exists R.C$ is replaced by the concept $\exists(R \sqcap Q_C) \sqcap \forall(R \sqcap Q_C).C$, adding Q_C as a new role name (a different Q_C for each occurrence of $\exists R.C$, to be used nowhere else). We call $\widetilde{D}$ an $\sqcap$-*simulation* of D in the rest of the chapter.

The proof that the simulation is correct can be easily given by referring to tableaux.

Example 3.8 Considering the concept D below on the left, and simulating qualified existential quantifications in D by role intersections, one obtains the concept $\widetilde{D}$ on the right,

$$D = \begin{cases} \exists R.A \sqcap \\ \exists R.B \sqcap \\ \forall R.C \end{cases} \qquad \widetilde{D} = \begin{cases} \exists(R \sqcap Q_A) \sqcap \forall(R \sqcap Q_A).A \sqcap \\ \exists(R \sqcap Q_B) \sqcap \forall(R \sqcap Q_B).B \sqcap \\ \forall R.C \end{cases}$$

where subscripts on new role names help identifying which existential they simulate. Applying tableaux rules of Chapter 2 to $\widetilde{D}(x)$, one obtains the model

$$\begin{array}{rl} R(x,y) & A(y) \\ Q_A(x,y) & C(y) \\ R(x,z) & B(z) \\ Q_B(x,z) & C(z) \end{array}$$

which satisfies both concepts. ■

Proposition 3.9 *A concept D is satisfiable iff $\widetilde{D}$ is satisfiable.*

Proof The proof of the proposition follows the example. Namely, an open tableau branch for $\widetilde{D}$ is also an open tableau branch for D (ignoring assertions on new role names), and an open tableau branch for D can be transformed to an open tableau branch for $\widetilde{D}$ just by adding the assertions about new role names. $\square$

As observed by Nebel [1990a], an acyclic role hierarchy in a description logic can be always simulated by conjunctions of existing roles and new role names. In the above example, using two role names Q_A, Q_B and the inclusions $Q_A \sqsubseteq R$, $Q_B \sqsubseteq R$ yields the same simulation.

Applying $\sqcap$-simulation, one could obtain from the reduction in Theorem 3.6 the original reduction by Nebel, proving that satisfiability (and non-subsumption) in $\mathcal{ALN}(\sqcap)$ is NP-hard. Using a more complex reduction, Donini *et al.* [1997a] proved that satisfiability in $\mathcal{ALN}(\sqcap)$ is in fact PSPACE-complete.

3.3 AND-branching: finding a clash

When candidate models of a concept have exponential size—as for the $\mathcal{ALE}$-concept of Section 3.1.1.2—models cannot be guessed and checked in polynomial time. In this case, a combinatorial problem is finding the clash—if any—in the candidate model. This leads to NP-hardness of *un*satisfiability and subsumption. However, for many DLs the AND-tree structure of a model is such that its traces (branches of the AND-tree) have polynomial size. A concept C is satisfiable iff there is no trace containing a clash, hence it is sufficient to guess such a trace to show that C is unsatisfiable. From this argument, Schmidt-Schauß and Smolka [1991] proved that satisfiability in $\mathcal{ALE}$ is in coNP.

3.3.1 Intractability of satisfiability in $\mathcal{ALE}$

We now report a proof that satisfiability in $\mathcal{ALE}$ is coNP-complete. The original proof was based on a polynomial-time reduction from a variant of the NP-complete problem ONE-IN-THREE 3SAT [Garey and Johnson, 1979, p. 259]. Here we present a proof based on the same idea, but with a slightly different construction, relying on a reduction from the NP-complete problem EXACT COVER (XC) [Garey and Johnson, 1979, p. 221]. Such a problem is defined as follows.

Definition 3.10 (Exact cover xc) Let $U = \{u_1, \ldots, u_n\}$ be a finite set, and let $\mathcal{M}$ be a family $M_1, \ldots, M_m$ of subsets of U. Decide if there are q mutually disjoint subsets $M_{i_1}, \ldots, M_{i_q}$ such that their union equals U, i.e., $M_{i_h} \cap M_{i_k} = \emptyset$ for $1 \leq h < k \leq q$, and $\bigcup_{k=1}^{q} M_{i_k} = U$. ■

The reduction consists in associating every instance of XC with an $\mathcal{ALE}$-concept $C_{\mathcal{M}}$, such that $\mathcal{M}$ has an exact cover if and only if $C_{\mathcal{M}}$ is *unsatisfiable*. It is important to note that, differently from the previous sections, here a solution of the NP-complete source problem is related to a proof of the *absence* of a model. In fact, exact covers of $\mathcal{M}$ are related to those traces of $\{C_{\mathcal{M}}(x)\}$ that contain a clash, hence the certificate of a solution of an NP-complete problem is related to a refutation in the target DL.

In the following we assume R to be a role name. We translate $\mathcal{M}$ into the concept

$$C_{\mathcal{M}} = C_1^1 \sqcap \cdots \sqcap C_1^m \sqcap D_1$$

where each concept C_1^j represents a subset M_j, and is inductively defined as

$$C_l^j = \begin{cases} \exists R.C_{l+1}^j, & \text{if either } l \leq n,\ u_l \in M_j \text{ or } l > n,\ u_{l-n} \in M_j \\ \forall R.C_{l+1}^j, & \text{if either } l \leq n,\ u_l \notin M_j \text{ or } l > n,\ u_{l-n} \notin M_j \end{cases} \quad \text{for } l \in \{1, \ldots, 2n\}$$

and by the base case $C_{2n+1}^j = \top$. The concept D_1 is defined by

$$D_1 = \underbrace{\forall R. \cdots \forall R.}_{2n} \bot$$

and each one of $D_2, D_3, \ldots$ have one universal quantifier less than the previous one.

Intuitively, for every element u_l in U there are two corresponding levels $l, l+n$ in the concepts C_1^j's, where "level" refers to the nesting of quantifiers. The element u_l is present in M_j if and only if there is an existential quantifier in the concept C_1^j at level $l+n$—which implies by construction that $\exists$ is also at level l. The concept D_1 is designed in such a way that a clash for $\{C_{\mathcal{M}}(x)\}$ can only occur in a trace containing at least $2n+1$ individual names.

Example 3.11 Consider the following instance of XC: let $U = \{u_1, \ldots, u_3\}$, and

$$\mathcal{M} = \{M_1 = \{u_1, u_2\}, M_2 = \{u_2, u_3\}, M_3 = \{u_3\}\}$$

The corresponding $\mathcal{ALE}$-concept $C_{\mathcal{M}}$ is given by the conjunction of C_1^1, C_1^2, C_1^3 and D_1, defined as follows.

$$\begin{array}{ccccc} & & & & \;\; u_1 \;\; u_2 \;\; u_3 \;\; u_1 \;\; u_2 \;\; u_3 \\ \hline M_1 & \leftrightarrow & C_1^1 & = & \exists R.\exists R.\forall R.\exists R.\exists R.\forall R.\top \\ M_2 & \leftrightarrow & C_1^2 & = & \forall R.\exists R.\exists R.\forall R.\exists R.\exists R.\top \\ M_3 & \leftrightarrow & C_1^3 & = & \forall R.\forall R.\exists R.\forall R.\forall R.\exists R.\top \\ & & D_1 & = & \forall R.\forall R.\forall R.\forall R.\forall R.\forall R.\bot \end{array}$$

where on the left we put the subset M_j corresponding to each C_1^j, and above the elements of U corresponding to each level of the concepts. Observe that the elements of U appear twice. ■

The conjunction of the above concepts is unsatisfiable if and only if the interplay of the various existential and universal quantifiers, represented by a trace, forces an individual name in the tableau for $\{C_{\mathcal{M}}(x)\}$ to belong to the extension of $\bot$. This reduction creates a correspondence between such a trace and an exact cover of U.

In order to formally characterize such a correspondence, we define the activeness of a concept in a trace. Let T be a trace and C be a concept. We say that C is *active in* T if C is of the form $\exists R.D$ and there are individual names y, z such that T contains $C(y)$, $R(y,z)$, and $D(z)$. Therefore, an existentially quantified concept $\exists R.D$ is active in T if the $\rightarrow_{\exists}$-rule has been applied to the assertion $\exists R.D(y)$ in T. Intuitively, if C_k^j is active in a trace of $\{C_{\mathcal{M}}(x)\}$ containing a clash, then u_k belongs to an exact cover of $\mathcal{M}$.

Lemma 3.12 ([Donini *et al.*, 1992a, Lemma 3.1]) *Let T be a trace of $\{C_{\mathcal{M}}(x)\}$.*

(i) *Suppose C_k^j is active in T. Then for all $l \in \{1, \ldots, k\}$ if the concept C_l^j is of the form $\exists R.C_{l+1}^j$, then it is active in T.*
(ii) *If T contains a clash, then for every $l \in \{1, \ldots, 2n\}$ there exists exactly one j such that C_l^j is active in T.*

Example 3.13 The reader can gain an insight on the importance of the above properties by constructing the tableau for the concept

$$\begin{array}{l}(\exists R.\forall R.\exists R.A) \sqcap \\ (\exists R.\forall R.\exists R.B) \sqcap \\ (\forall R.\exists R.\top)\end{array}$$

and verifying that the trace reaching the concept A has both existentials in the first line active (and no existential of the second line), and vice versa for the trace reaching B. ■

Example 3.14 (Example 3.11 continued) Note that in Example 3.11 the two subsets M_1 and M_2 form a (non-exact) cover of U, and indeed, the tableau for $\{C_1^1 \sqcap C_2^1 \sqcap D_1(x)\}$ is satisfiable. Moreover, observe the importance of the two levels. If concepts were formed by just one level, the following concepts would be unsatisfiable (choose highlighted existentials):

$$\begin{array}{rcl}\overline{C_1^1} & = & \exists \mathbf{R}.\exists R.\forall R.\top \\ \overline{C_2^1} & = & \forall R.\exists \mathbf{R}.\exists \mathbf{R}.\top \\ \overline{D_1} & = & \forall R.\forall R.\forall R.\bot\end{array}$$

corresponding to a cover by M_1 and M_2 which is non-exact. The second level ensures that once an existential is chosen, all nested existentials must be chosen too to form a trace. ■

Theorem 3.15 *Unsatisfiability in $\mathcal{ALE}$ is* NP*-hard.*

Proof We show that an instance $(U, \mathcal{M})$ of XC has an exact cover if and only if $C_{\mathcal{M}}$ is unsatisfiable. Let $\mathcal{M} = \{M_1, \ldots, M_m\}$ be a set of subsets from U and $C_{\mathcal{M}} = C_1^1 \sqcap \ldots \sqcap C_1^m \sqcap D_1$ be the corresponding concept. Since this proof is the base for three other ones in the chapter, we present it with some detail.

Only-if Let $M_{i_1}, \ldots, M_{i_q}$ be an exact cover of U. Let T be a trace of $\{C_{\mathcal{M}}(x_1)\}$ defined inductively as follows:

$$T_1 = \{C_1^j(x_1) \mid j \in \{1, \ldots, m\}\} \cup \{D_1(x_1)\}$$

$$T_{l+1} = T_l \cup \{R(x_l, x_{l+1})\} \cup \{C_{l+1}^j(x_{l+1}) \mid u_{l+1} \in M_j\} \cup \{D_{l+1}(x_{l+1})\}$$

Obviously, $T = T_{2n+1}$ contains a clash, because $D_{2n+1} = \bot$. For each level l there is exactly one j such that $C_l^j = \exists R.C_{l+1}^j$. Using this fact, one can easily show that T is a trace by induction on l.

If If $C_{\mathcal{M}}$ is unsatisfiable, then there exists a trace T of $\{C_{\mathcal{M}}(x)\}$ such that T contains a clash. We show that the subsets in

$$\{M_j \mid \exists l \in \{1, \ldots, n\} : C_{n+l}^j \text{ is active in } T\}$$

form an exact cover of U. First of all, since T is a trace, for every level $l \in \{1, \ldots, 2n\}$ there exists a j such that C_l^j is active in T (second point of Lemma 3.12). Hence the union of these subsets cover U.

We now prove that no two subsets overlap: in fact, suppose there are i, j such that M_i, M_j intersect non-trivially in element u_l. Here we exploit the two-layered construction of $C_{\mathcal{M}}$. By definition, there are h, k such that C_{n+h}^i and C_{n+k}^j are active in T. Since u_l is in both M_i and M_j, by construction of $C_{\mathcal{M}}$ we have $C_l^i = \exists R.C_{l+1}^i$ and $C_l^j = \exists R.C_{l+1}^j$. From first point in Lemma 3.12, we know that C_l^i and C_l^j are both active in T. Hence $i = j$ from second point of Lemma 3.12. □

The above reduction works also for the special case of XC in which every subset has at most three elements, which corresponds to at most six nested existential quantifications in each concept C_1^j. Hence, bounding by a constant $k \geq 6$ the number of nested existential quantifications does not yield tractability. The original reduction from ONE-IN-THREE 3SAT shows that also bounding by a constant $k \geq 3$ the number of existentials in each level, does not yield tractability.

Simulating qualified existential quantifications in $C_{\mathcal{M}}$ by role intersection (see Section 3.2.2.1), we conclude that unsatisfiability of concepts in $\mathcal{AL}(\sqcap)$—$\mathcal{AL}$ plus role conjunction—is NP-hard, too.

Theorem 3.16 *Satisfiability and subsumption of concepts are* NP*-hard in* $\mathcal{AL}(\sqcap)$*.*

We note that this source of intractability is not due to the presence of the concept $\bot$, but to the interplay of universal and existential quantification. In fact, the above reduction works also for the description logic $\mathcal{FL}^-\mathcal{E}$, which is $\mathcal{FL}^-$ plus qualified existential quantification.

Theorem 3.17 *Subsumption is* NP*-hard in* $\mathcal{FL}^-\mathcal{E}$*.*

Proof The proof is based on the reduction given for $\mathcal{ALE}$. The $\mathcal{ALE}$-concept $C_{\mathcal{M}} = C_1^1 \sqcap \ldots \sqcap C_1^m \sqcap D_1$ in that reduction, is unsatisfiable if and only if $C_1^1 \sqcap \ldots \sqcap C_1^m$ is subsumed by $\neg D_1$. Now $C_1^1 \sqcap \ldots \sqcap C_1^m$ is a concept in $\mathcal{FL}^-\mathcal{E}$ and $\neg D_1$ can be rewritten to the equivalent concept E, defined as

$$E = \underbrace{\exists R. \cdots \exists R.}_{2n} \top$$

i.e., a chain of $2n$ qualified existential quantifications terminating with the concept $\top$. Obviously, E is in $\mathcal{FL}^-\mathcal{E}$, hence subsumption in $\mathcal{FL}^-\mathcal{E}$ is NP-hard. $\square$

We now use the above construction to show that in three other DLs—extending $\mathcal{FL}^-$ with each pair of role constructs for role conjunction, role inverse, and role chain—subsumption is NP-hard. The fact that reductions can be easily reused is a characteristic of DLs. It depends on the compositional semantics of constructs—hardness proofs obviously carry over to more general DLs—but also on the extensional semantics, that allows one to simulate a construct with others.

3.3.2 $\mathcal{FL}^-$ plus role conjunction and role inverse

We abbreviate this description logic as $\mathcal{FL}^-(\sqcap,^-)$. We prove that $\mathcal{FL}^-(\sqcap,^-)$ is hard for NP with an argument similar to that for $\mathcal{FL}^-\mathcal{E}$. One may be tempted to use $\sqcap$-simulation, defined in Section 3.2.2.1, which substitutes qualified existential quantifications with role intersections. However, a direct $\sqcap$-simulation of the concepts used in the reduction for $\mathcal{FL}^-\mathcal{E}$ does not work. In fact, $\sqcap$-simulation preserves satisfiability, not subsumption; e.g., while $\exists R.C \sqcap D$ is subsumed by $\exists R.C$, its $\sqcap$-simulation $\exists(R \sqcap Q_1) \sqcap \forall Q_1.C \sqcap D$ is not subsumed by $\exists(R \sqcap Q_2) \sqcap \forall Q_2.C$.

To carry over the proof, it is useful a tableaux rule for role inverse:

Condition: $\mathcal{T}$ contains $R(x, y)$,
where R is either a role name P or its inverse P^-;
Action: $\mathcal{T}' = \mathcal{T} \cup \{R^-(y, x)\}$,
where if $R = P^-$, then $R^- = P$.

Theorem 3.18 *Subsumption in $\mathcal{FL}^-(\sqcap,^-)$ is* NP*-hard.*

Proof We refer to the concept $C_{\mathcal{M}}$ defined in the reduction given for $\mathcal{ALE}$. Let n be the cardinality of U in XC. First define the concept F as follows:

$$F = \underbrace{\forall R.\cdots\forall R.}_{2n}\underbrace{\forall(R^-).\cdots\forall(R^-).}_{2n}A$$

where A is a concept name (remind that $C_{\mathcal{M}}$ does not contain any concept name, but $\top$ and $\bot$). F is a concept of $\mathcal{FL}^-(\sqcap,^-)$.

Observe now that the $\mathcal{ALE}$-concept $C_{\mathcal{M}} = C_1^1 \sqcap \ldots \sqcap C_1^m \sqcap D_1$ is unsatisfiable if and only if $\widetilde{C}_1^1 \sqcap \ldots \sqcap \widetilde{C}_1^m \sqcap F$ is subsumed by A (where $\widetilde{C}$ is the $\sqcap$-simulation of C). In fact, the subsumption holds if and only if the complete tableau for $\{\widetilde{C}_1^1 \sqcap \ldots \sqcap \widetilde{C}_1^m \sqcap F(x), \neg A(x)\}$ contains the only possible clash $\{A(x), \neg A(x)\}$. This tableau contains a clash if and only if there is a trace of length $2n$ in the tableau, and such a trace is in one-one correspondence with the exact covers of the problem XC. Hence subsumption in $\mathcal{FL}^-(\sqcap,^-)$ is NP-hard. □

3.3.3 $\mathcal{FL}^-$ plus role conjunction and role chain

We abbreviate this description logic as $\mathcal{FL}^-(\sqcap, \circ)$.

Theorem 3.19 *Subsumption in $\mathcal{FL}^-(\sqcap, \circ)$ is* NP*-hard.*

Proof Again, we refer to the concept $C_{\mathcal{M}}$ defined in the reduction given for $\mathcal{ALE}$. Observe that the $\mathcal{ALE}$-concept $C_{\mathcal{M}} = C_1^1 \sqcap \ldots \sqcap C_1^m \sqcap D_1$ is unsatisfiable if and only if $\widetilde{C}_1^1 \sqcap \ldots \sqcap \widetilde{C}_1^m$ is subsumed by $\neg D_1$ (again, $\widetilde{C}$ is the $\sqcap$-simulation of C). The claim holds, since $\widetilde{C}_1^1 \sqcap \ldots \sqcap \widetilde{C}_1^m$ is in $\mathcal{FL}^-(\sqcap)$ and $\neg D_1$ can be expressed as the equivalent concept E, defined as follows:

$$G = \exists\underbrace{(R \circ \cdots \circ R)}_{2m} \tag{3.8}$$

Obviously, G is in $\mathcal{FL}^-(\circ)$, hence subsumption in $\mathcal{FL}^-(\sqcap, \circ)$ is NP-hard. □

We note that in the above reduction, subsumption is proved intractable by using only role conjunction in the subsumee (to simulate existential quantification), and only role chain in the subsumer. We exploit this fact in the following section.

3.3.4 $\mathcal{FL}^-$ plus role chain and role inverse

We abbreviate this description logic as $\mathcal{FL}^-(\circ,^-)$. We first show that, similarly to Section 3.2.2.1, qualified existential quantifications in a concept D can be replaced by a combination of role chains and role inverses, obtaining a new concept $\widehat{D}$ that is satisfiable iff D does.

3.3.4.1 Simulating $\exists R.C$ via role chains and role inverses

Donini *et al.* [1991b; 1999] showed that a concept D containing qualified existential role quantifications $\exists R.C$ is satisfiable iff the concept $\widehat{D}$ is satisfiable, where in $\widehat{D}$ each occurrence of a concept $\exists R.C$ is replaced by the concept $\exists(R \circ Q_C) \sqcap \forall(R \circ Q_C \circ Q_C^-).C$, adding Q_C as a new role name (a different Q for each occurrence of $\exists R.C$, to be used nowhere else). We say that $\widehat{C}$ is a $\circ$-*simulation* of C.

Also this simulation can be explained by referring to tableaux, through an example concept.

Example 3.20 Consider the concept D below on the left, and its $\circ$-simulation $\widehat{D}$ on the right:

$$D = \begin{cases} \exists R.A \sqcap \\ \exists R.B \sqcap \\ \forall R.C \end{cases} \qquad \widehat{D} = \begin{cases} \exists(R \circ Q_A) \sqcap \forall(R \circ Q_A \circ Q_A^-).A \sqcap \\ \exists(R \circ Q_B) \sqcap \forall(R \circ Q_B \circ Q_B^-).B \sqcap \\ \forall R.C \end{cases}$$

where subscripts on new role names help identifying which existential they simulate. Applying tableau rules of Chapter 2 to $\widehat{D}(x)$, one obtains the model

$$\begin{array}{lll} R(x,y) & A(y) & Q_A(y,u_y) \\ & C(y) & \\ R(x,z) & B(z) & Q_B(z,u_z) \\ & C(z) & \end{array}$$

where subscripts on individuals u_y, u_z highlight that there is a new individual name for each individual name used to satisfy an existential quantification. That is, the number of individual names in the tableau for $\widehat{D}$ are at most twice those in the tableau for D. ∎

Lemma 3.21 *Let C be an $\mathcal{ALE}$-concept and $\widehat{C}$ its $\circ$-simulation. Then C is satisfiable if and only if $\widehat{C}$ is satisfiable.*

Proof The proof extends the above example. In one direction, an open tableau for $\widehat{D}$ is also an open tableau for D (ignoring assertions on new role names). In the other direction, an open tableau for D can be transformed to an open tableau for $\widetilde{D}$: to every role assertion $R(x,y)$—added to satisfy an existential $\exists R.C$ in D—chain an assertion $Q_C(y,u_y)$. □

If C is an $\mathcal{ALE}$-concept, its $\circ$-simulation $\widehat{C}$ is a concept belonging to the language $\mathcal{AL}(\circ,^-)$, that is, $\mathcal{AL}$ plus role inverses and role chains. Of course, $\circ$-simulations could be defined for concepts belonging to DLs more expressive than $\mathcal{ALE}$. For DLs in which every concept is satisfiable (like $\mathcal{FL}^-(\circ,^-)$) this simulation can be interesting only in subsumptions.

We can now come back to subsumption in the DL $\mathcal{FL}^-$ plus role inverses and role chains.

Theorem 3.22 *Subsumption in $\mathcal{FL}^-(\circ,^-)$ is* NP*-hard.*

Proof For every $\mathcal{ALE}$-concept C, one can compute in quadratic time an $\circ$-simulation $\widehat{C}$. For a given instance $(U, \mathcal{M})$ of XC, $C_{\mathcal{M}}$ is unsatisfiable iff (by Lemma 3.21) $\widehat{C}_{\mathcal{M}}$ is satisfiable iff $\widehat{C}_1^1 \sqcap \ldots \sqcap \widehat{C}_1^m$ is subsumed by $\neg D_1$. Now the subsumee contains no negated concept, hence it belongs to $\mathcal{FL}^-(\circ,^-)$. The subsumer is equivalent to the concept G in (3.8), which again is in $\mathcal{FL}^-(\circ,^-)$. □

3.4 Combining sources of complexity

In a DL containing both sources of complexity, one might expect to code any problem involving the exploration of polynomial-depth, rooted AND-OR graphs. The computational analog of such graphs is the class APTime (problems solved in polynomial time by an alternating Turing machine) which is equivalent to PSpace (e.g., see [Johnson, 1990, p. 98]). A well-known PSpace-complete problem is Validity of Quantified Boolean Formulae:

Definition 3.23 (Quantified Boolean Formulae QBF) Decide whether it is valid the (second-order logic) closed sentence

$$(Q_1 X_1)(Q_2 X_2)\cdots(Q_n X_n)[F(X_1,\ldots,X_n)]$$

where each Q_i is a quantifier (either $\forall$ or $\exists$) and $F(X_1,\ldots,X_n)$ is a Boolean formula with Boolean variables $X_1,\ldots,X_n$. ■

The problem remains PSpace-complete if F is in 3CNF, i.e., conjunctive normal form with at most three literals per clause. We call *prefix* of the quantified formula the string of quantifiers, and *matrix* the 3CNF formula F.

This problem can be encoded in an AND-OR graph, using AND-nodes to encode $\forall$-quantifiers, and OR-nodes for $\exists$-quantifiers. In the leaves, there is the matrix F. We use this analogy to illustrate the reduction, taken from [Schmidt-Schauß and Smolka, 1991].

3.4.1 PSPACE -hardness of satisfiability in $\mathcal{ALC}$

Without loss of generality, we assume that each clause is non-tautological, i.e., a literal and its complement do not appear both in the same clause. Let $F = G_1 \wedge \cdots \wedge G_m$. The QBF $(Q_1X_1)\cdots(Q_nX_n)[G_1 \wedge \cdots \wedge G_m]$ is valid iff the $\mathcal{ALC}$-concept

$$C = D \sqcap C_1^1 \sqcap \ldots \sqcap C_1^n \tag{3.9}$$

is satisfiable, where in C all concepts are formed using the concept name A and the atomic role name R. The concept D encodes the prefix, and is of the form $D_1 \sqcap \forall R.(D_2 \sqcap \forall R.(\ldots(D_{n-1} \sqcap \forall R.D_n)\ldots)$ where for $i \in \{1,\ldots,n\}$ each D_i corresponds to a quantifier of the QBF in the following way:

$$D_i = \begin{cases} (\exists R.A) \sqcap (\exists R.\neg A), & \text{if } Q_i = \forall \\ \exists R.\top, & \text{if } Q_i = \exists \end{cases}$$

The concept C_1^i is obtained from the clause G_i using the concept name A when a Boolean variable occurs positively in G_i, $\neg A$ when it occurs negatively, and nesting l universal role quantifications to encode the variable X_l. In detail, let k be the maximum index of all Boolean variables appearing in G_i. Then, for $l \in \{1,\ldots,(k-1)\}$ one defines

$$C_l^i = \begin{cases} \forall R.(A \sqcup C_{l+1}^i), & \text{if } X_l \text{ appears positively in } G_i \\ \forall R.(\neg A \sqcup C_{l+1}^i), & \text{if } X_l \text{ appears negatively in } G_i \\ \forall R.C_{l+1}^i, & \text{if } X_l \text{ does not appear in } G_i \end{cases}$$

and the last concept of the sequence is defined as

$$C_k^i = \begin{cases} \forall R.A, & \text{if } X_k \text{ appears positively in } G_i \\ \forall R.\neg A, & \text{if } X_k \text{ appears negatively in } G_i \end{cases}$$

It can be shown that each trace in a tableau branch for D corresponds to a truth assignment to the Boolean variables, and that all traces of a tableau branch correspond to a set of truth assignments consistent with the prefix. Therefore, Schmidt-Schauß and Smolka conclude that satisfiability in $\mathcal{ALC}$ is PSPACE-hard. Combining this result with the polynomial-space calculus given for $\mathcal{ALCN}$ in Chapter 2, one obtains that satisfiability (and subsumption) in $\mathcal{ALCN}$ are PSPACE-complete, and that the exponential-time behavior of the calculus cannot be improved unless PSPACE =PTIME. Satisfiability and subsumption are still in PSPACE if role conjuctions are added to $\mathcal{ALCN}$ [Donini *et al.*, 1997a], or if inverse roles and transitive roles are added to $\mathcal{ALC}$ [Horrocks *et al.*, 2000b].

Using $\sqcap$-simulations, one can use the same reduction to prove that both satisfiability and subsumption in $\mathcal{ALU}(\sqcap)$ are PSPACE-hard (and thus PSPACE-complete). With a more complex reduction, Donini *et al.* [1991a] proved that also satisfiability in $\mathcal{ALN}(\sqcap)$ is PSPACE-hard. Hemaspaandra [1999] proved that satisfiability in

$\mathcal{ALEN}$ is PSpace-hard using a reduction from QBF, where the prefix was coded with a concept similar to D (more precisely, similar to the concept D in Section 3.1.1.2), and the matrix was coded in a more complex way. Also $\mathcal{FL}$ was proved PSpace-hard in [Donini *et al.*, 1997a]. Observe that all these DLs contain both sources of complexity.

3.4.2 A remark on reductions

Schild [1991] observed that $\mathcal{ALC}$ is a notational variant of multi-modal logic **K**, whose satisfiability was proved PSpace-hard by Ladner [1977], using a different reduction from QBF. This gives us the occasion to point out a characteristic of reductions from a different, pretty experimental viewpoint.

The target modal formula in Ladner's reduction has size quadratic w.r.t. the given instance of QBF, while one can observe that the concept C in (3.9) has just linear size. From a theoretical perspective of the PSpace reduction, this is irrelevant. However, QBF has been studied also from an experimental point of view (e.g., [Cadoli *et al.*, 2000; Gent and Walsh, 1999]): trivial cases have been identified, easy-hard-easy patterns have been found, and one can use ratios of clauses/variables for which the probability that a random QBF is valid is around 0.5—which have been proved experimentally to contain the "hard" instances. This experimental work can be transferred in DLs, to compare the various algorithms and systems for reasoning in $\mathcal{ALC}$. This transfer yields the benefits that

- concepts which are trivially (un)satifiable do not need to be isolated again;
- the translation of "hard" QBFs can be used to test reasoning algorithms for $\mathcal{ALC}$;
- the performance of algorithms for $\mathcal{ALC}$ can be compared with best known algorithms for solving QBF(see [Cadoli *et al.*, 2000; Rintanen, 1999; Giunchiglia *et al.*, 2001b]), and optimizations can be carried over.

However, using Ladner's reduction to obtain "hard-to-reason" concepts, the quadratic blow-up of the reduction makes the resulting concepts soon too big to be significantly tested. Using Schmidt-Schauß and Smolka linear reduction, instead, one can use a spectrum of "hard" concepts as wide as the original instances of QBF. Thus, experimental analysis might make significant differences between (theoretically equivalent) polynomial many-one transformations used in reductions [Donini and Massacci, 2000].

3.5 Reasoning in the presence of axioms

In this section we consider the impact of axioms on reasoning. Intuitively, axioms introduce new concept expressions in every individual generated in a tableau, hence simple arguments on termination and complexity based on the nesting of operators do not apply. We start with a comparison with Dynamic Logic, and then we show how axioms can encode a succinct representation of AND-OR graphs, leading to an ExpTime lower bound.

3.5.1 Results from Propositional Dynamic Logic

Propositional Dynamic Logic (PDL) [Harel *et al.*, 2000] is a formalism able to express propositional properties of programs. Instead of introducing yet another logical syntax, we will talk about PDL in terms of DLs. A precise correspondence between DLs and PDL can be found in Chapter 5.

The counterpart of PDL in DLs is $\mathcal{ALC}_{trans}$ [Baader, 1991], already defined in Chapter 2. We recall that $\mathcal{ALC}_{trans}$ is $\mathcal{ALC}$ plus a rich set of role constructors: union of roles, composition, and transitive closure. To be precise, PDL has also a role-forming constructor which is role identity, and the closure of a role is the reflexive-transitive one, denoted as R^*. Reflexive-transitive closure is defined similarly to transitive closure, but considering also every pair (a, a) is in the interpretation of R^*. However, Schild [1991] showed that these are minor differences, as far as we are concerned with computational behavior only.

PDL and $\mathcal{ALC}_{trans}$ are relevant in this section about axioms, because using union and transitive closure of roles, one can "internalize" axioms in a concept in the following way [Baader, 1991; Schild, 1991]. Let C be an $\mathcal{ALC}$ concept, $\mathcal{T}$ a set of axioms of the form $C_i \sqsubseteq D_i$, $i \in \{1, \ldots, m\}$. Observe that every axiom can also be thought as a concept $\neg C \sqcup D$ which every individual in a model must belong to. Let $R_1, \ldots, R_n$ be all the role names used in either C or $\mathcal{T}$. Then C is satisfiable w.r.t. $\mathcal{T}$ iff the following concept is satisfiable:

$$C \sqcap \forall(R_1 \sqcup \cdots \sqcup R_n)^*.((\neg C_1 \sqcup D_1) \sqcap \cdots \sqcap (\neg C_m \sqcup D_m)) \tag{3.10}$$

The key property that makes this reduction correct is the connected model property [Streett, 1982]: if C has a model w.r.t. a set of axioms, then it has also a model in which one element $a \in \Delta^{\mathcal{I}}$ is in $C^{\mathcal{I}}$, and for every other element b in the model, there is a path of roles from a to b.

Concept (3.10) is just a syntactic variant of a PDL expression. Hence, every upper bound on complexity of satisfiability for PDL applies also to concept satisfiability in $\mathcal{ALC}$ w.r.t. axioms, including all role constructors of PDL. Namely, satisfiability in PDL was proved to be decidable in deterministic exponential time, first by Pratt [1979], and then by Vardi and Wolper [1986] using an embedding into tree automata. This upper bound holds also for $\mathcal{ALC}$ plus axioms. It is interesting to observe that the deterministic exponential time upper bound was nontrivial; simple nondeterministic upper bounds were proved by Fischer and Ladner [1979] for PDL and by Buchheit *et al.* [1993a] for DLs, using tableaux. Only recently a tableaux with lemmata providing a deterministic exponential upper bound has been found [Donini and Massacci, 2000].

Regarding hardness, every lower bound on reasoning in $\mathcal{ALC}$ with axioms carries over to PDL. However, lower bounds for PDL were already known. Fischer and Ladner [1979] proved that PDL is ExpTime-hard using a reduction from Alternating Turing Machines working in polynomial space (recall that the complexity class Alternating Polynomial Space is the same as ExpTime [Johnson, 1990]). van Emde Boas [1997] proved the same result using a reduction from alternating domino games. However, both hardness proofs use a very small part of PDL, and in particular, transitive closure on roles appears only in one expression of the form (3.10), so that proofs could be adapted to $\mathcal{ALC}$ concept satisfiability w.r.t. a set of inclusions, in a very simple way. Moreover, the proofs use $\forall R.C$ to code an AND-node, and $\exists R.C$ to code an OR-node. Hence, they follow the same intuition presented in the previous section, where we showed the correspondence between AND-OR-trees and satisfiability of $\mathcal{ALC}$ without axioms.

Here, we want to present yet another proof, of a very different nature, that highlights the fact that concept inclusions can express a large structure in a succinct way.

3.5.2 Axioms and succinct representations of AND-OR-graphs

We now need more precise definitions about AND-OR-graphs. An AND-OR-graph is a graph in which nodes are partitioned into AND-nodes, and OR-nodes. An OR-node is reachable if one of its predecessors is reachable (as in ordinary graphs), while an AND-node is reachable only if all its predecessors are reachable.

Definition 3.24 (AND-OR-Graph Accessibility Problem (AGAP)) Given an AND-OR-graph, a set of source nodes $S_1, \ldots, S_m$, and a target node T, is T reachable from $S_1, \ldots, S_m$? ■

Let n be the number of nodes of the graph, and d (a constant) the maximum number of predecessors of a node. It is well known that AGAP can be solved in time polynomial in n (e.g., it can be reduced to Monotone Circuit Value, which is PTime-complete [Papadimitriou, 1994]). However, AGAP becomes ExpTime-complete when one considers its succinct version [Balcazar, 1996]. Let the out-degree of a node be bounded by a constant d. Let $\mathbf{C}$ be a Boolean circuit with $\log n$ inputs, and with $1 + d \log n$ outputs; when the input of $\mathbf{C}$ is the binary encoding of a node N, its outputs are the encodings of the type of N (AND/OR) and of the d predecessors of N (using a dummy node if the predecessors are less than d).

Definition 3.25 (Succinct AND-OR-Graph Accessibility Problem (s(AGAP))) Given a circuit $\mathbf{C}$ representing an AND-OR-graph, a set of source nodes $S_1, \ldots, S_m$, and a target node T, is T reachable from $S_1, \ldots, S_m$? ■

Now, s(AGAP) is EXPTIME-complete [Balcazar, 1996]. The intuition for this exponential blow-up in complexity is that there are many circuits which can encode graphs whose size is exponentially larger than the circuit size. This intuition applies to many other succinct representations of problems with circuits [Papadimitriou, 1994, p. 492] or with propositional formulae [Veith, 1997], yielding complete problems for high complexity classes.

We reduce s(AGAP) for graphs with in-degree $d = 2$ to unsatisfiability of an $\mathcal{ALC}$ concept C w.r.t. a set of inclusions $\mathcal{T}$. Intuitively, the axioms can succinctly encode either a proof of unsatisfiability for a concept, or a model for C w.r.t. $\mathcal{T}$. We note that, since we are coding reachability into unsatisfiability, we will use $\sqcap$ to code OR-nodes—a conjunction is unsatisfiable when at least one of its conjuncts does—and $\sqcup$ to code AND-nodes.

First of all, let $A_1, \ldots, A_{\log n}$, be a set of concept names one-one with the inputs of the circuit $\mathbf{C}$. Each node N in the graph is then mapped into a conjunction of As and their negations, denoted as *concept*(N), depending on the code of N: if the i-th bit in the code of N is 1, use A_i, if it is 0, use $\neg A_i$. For example, if N has code 1101 then *concept*(N) is $A_1 \sqcap A_2 \sqcap \neg A_3 \sqcap A_4$.

Then, let $B^1_1, \ldots, B^1_{\log n}$, and $B^2_1, \ldots, B^2_{\log n}$ be two sets of concept names one-one with the outputs of $\mathbf{C}$. Conjunctions of Bs with negations code predecessor nodes.

Moreover, let two concept names AND, OR, represent the type of a graph node. If $\mathbf{C}$ has k internal gates, we use also k concept names $W_1, \ldots, W_k$. For each gate, we use a concept equality that mimics the Boolean formula defining the gate. E.g., if $\mathbf{C}$ has a $\wedge$-gate $x_1 \wedge x_2 = x_3$, we use the equality $X_1 \sqcap X_2 = X_3$, where the X_1, X_2, X_3 can be either concept names among $W_1, \ldots, W_k$ denoting input/output of internal gates, or they can be some of the As and Bs, denoting inputs/outputs of the whole circuit.

For the output of $\mathbf{C}$ encoding the type of the node, we use directly the two concept names AND, OR in the concept equality coding the output gate of $\mathbf{C}$. Moreover, to model the different interpretation of predecessors for the two type of nodes, we use the inclusions:

$$AND \sqsubseteq \exists R^1.\top \sqcup \exists R^2.\top \tag{3.11}$$
$$OR \sqsubseteq \exists R^1.\top \sqcap \exists R^2.\top \tag{3.12}$$

where R^1 and R^2 are two role names (we use indexes 1,2 to parallel indexes of the Bs). Observe that concept AND implies a disjunction $\sqcup$, and concept OR implies a conjunction $\sqcap$. This is because we reduce reachability to unsatisfiability, as we said before. Moreover, observe that *predecessors* in the AND-OR-graph are coded into role *successors* in the target DL.

For the output of $\mathbf{C}$ encoding the predecessors of a node, For $i \in \{1, \dots, \log n\}$, we add the following inclusions:

$$B_i^1 \sqsubseteq \forall R^1.A_i \tag{3.13}$$

$$\neg B_i^1 \sqsubseteq \forall R^1.\neg A_i \tag{3.14}$$

$$B_i^2 \sqsubseteq \forall R^2.A_i \tag{3.15}$$

$$\neg B_i^2 \sqsubseteq \forall R^2.\neg A_i \tag{3.16}$$

We denote by $\mathcal{T}_\mathbf{C}$ the set of all of the above axioms.

We now give an example of what the axioms imply. Suppose $\mathbf{C}$ computes the two predecessors 1011 and 0110 for node 1101. Then, equalities coding $\mathbf{C}$ force $\mathit{concept}(1101) = A_1 \sqcap A_2 \sqcap \neg A_3 \sqcap A_4$ to be included in B_1^1, $\neg B_2^1$, B_3^1, B_4^1 (first predecessor) and $\neg B_1^2$, B_2^2, B_3^2, $\neg B_4^2$ (second predecessor). Then inclusions (3.13)–(3.16) tell that every R^1-successor is included in A_1, $\neg A_2$, A_3, A_4—which conjoined, make *concept*(1011)—and that every R^2-successor is included in $\neg A_1$, A_2, A_3, $\neg A_4$ (*concept*(0110)). Moreover, if $\mathbf{C}$ computes an AND-type for node 1101, then axiom (3.11) implies that the corresponding concept is included in *AND*, and this implies that either an R^1-successor, or an R^2-successor exists. For OR-type nodes, both successors exist.

Theorem 3.26 *Let $\mathbf{C}$ be a circuit, T be the target node, and $S_1, \dots, S_m$ be the source nodes in an instance of s(*AGAP*). Then T is reachable from $S_1, \dots, S_m$ iff concept(T) is unsatisfiable in the TBox $\mathcal{T}_\mathbf{C} \cup \{\mathit{concept}(S_1) \sqsubseteq \bot\} \cup \cdots \cup \{\mathit{concept}(S_m) \sqsubseteq \bot\}$.*

Proof Most of the rationale of the proof has been informally given above. We sketch what is needed to complete the proof.

If Suppose T is unreachable from $S_1, \dots, S_m$. We construct a model $(\mathcal{I}, \Delta^\mathcal{I})$ for *concept*(T) satisfying the axioms as follows. Let $\Delta^\mathcal{I}$ be the set of all nodes in the graph which are unreachable from $S_1, \dots, S_m$. Then, $(R^1)^\mathcal{I}$ is the set of pairs (a, b) of nodes in $\Delta^\mathcal{I}$, such that b is the first predecessor of a, and similarly for $(R^2)^\mathcal{I}$ (second predecessor). For $i \in \{1, \dots, \log n\}$, $(A_i)^\mathcal{I}$ is the set of nodes in $\Delta^\mathcal{I}$ whose binary code has the i-th bit equal to 1. The interpretation of the Bs, Ws, *AND*, *OR*, concepts is according to the 1-value of the circuit: node a is in their interpretation iff the output they correspond to is 1 when the code of a is the input of the circuit.

Then, $T \in (\mathit{concept}(T))^\mathcal{I}$, and moreover $(\mathcal{I}, \Delta^\mathcal{I})$ satisfies by construction all axioms in $\mathcal{T}_\mathbf{C}$; e.g., if an OR-node is unreachable, then both its predecessors are unreachable, hence both predecessors are in $\Delta^{\mathcal{I}}$, and axiom (3.12) is satisfied. Similarly for an AND-node.

Only-if Let N be any node reachable from $S_1, \ldots, S_m$, and let $d(N)$ be the depth of the shortest hyperpath leading from $S_1, \ldots, S_m$ to N. We show by induction on $d(N)$ that *concept*(N) is unsatisfiable in the TBox.

If $d(N) = 0$, the claim holds by construction. Let N be a reachable node, with $d(N) = k + 1$. If N is an OR-node, at least one of its predecessors—let it be the first predecessor, and call it M—is reachable with $d(M) = k$. Then *concept*(M) is unsatisfiable by inductive hypothesis. But axiom (3.12) implies that *concept*(N) is included in $\exists R^1.\top \sqcap \exists R^2.\top$, while (3.13)–(3.16) imply that *concept*(N) is included in $\forall R^1.\mathit{concept}(M)$, that is, $\forall R^1.\bot$. Hence, also *concept*(N) is unsatisfiable. A similar proof holds in case N is an AND-node.

Then, the claim holds for $N = T$. □

Observe that in the above proof we did not use qualified existential quantification, hence, the proof works for the sublanguage of $\mathcal{ALC}$ called $\mathcal{ALU}$. Now, axioms coding the circuit can be propositionally rewritten without union. Moreover, the only other axiom in which union is needed is (3.11), which could be rewritten equivalently as $\forall R^1.\bot \sqcap \forall R^2.\bot \sqsubseteq \neg OR$, which is now in the language $\mathcal{AL}$.

Theorem 3.27 *Let C be a concept and $\mathcal{T}$ a set of inclusions in $\mathcal{AL}$, with at least two role names. Deciding whether C is unsatisfiable w.r.t. $\mathcal{T}$ is* ExpTime*-hard.*

The above theorem sharpens a result by Calvanese [1996b], who proved ExpTime-hardness for $\mathcal{ALU}$. McAllester *et al.* [1996] proved ExpTime-hardness for a logic that includes $\mathcal{FL}^{-}\mathcal{E}$, and their proof can be rewritten to work with $\mathcal{ALU}$.

We close the section with some discussion about the proof.

Remark 3.28 The above proof does not follow the correspondence used by Fischer and Ladner [1979] between AND-nodes and $\forall R.C$ concepts on one side, and OR-nodes and $\exists R.C$ concepts on the other side. Here, quantifications $\exists R$ and $\forall R.C$ were used to code predecessors in the graph, node type was coded by $\sqcap$, $\sqcup$ constructors, while axioms were crucial to mimic the behavior of the circuit. ■

3.5.3 Syntax restrictions on axioms

In the proof, no restriction on axioms was imposed. A significant syntactic restriction is to allow one to use only concept names on the left-hand side of axioms. In this case, a dependency graph induced by the axioms of a TBox $\mathcal{T}$ can be constructed, whose nodes are labeled by concept names. A node A is connected to a node B if the concept name B appears (also as a subconcept) in a concept C, and

$A \sqsubseteq C$ is an axiom. Then, it makes sense to distinguish between *cyclic* axioms, in which the dependency graph contains a cycle, and *acyclic* axioms.

Acyclicity is significant, because if only acyclic axioms are allowed, then reasoning in $\mathcal{ALC}$ can be performed in PSpace by expanding axioms when needed [Baader and Hollunder, 1991b; Calvanese, 1996b]. The only case for $\mathcal{ALC}$ (till now) in which acyclic axioms make reasoning ExpTime-hard is when concrete domains are also added [Lutz, 2001b].

Also sublanguages of $\mathcal{ALC}$ can be considered. With regard to acyclic axioms in $\mathcal{AL}$, Buchheit *et al.* [1998] proved that subsumption in acyclic $\mathcal{AL}$ TBoxes is coNP-hard, and in PSpace. Calvanese [1996b] proved that cyclic axioms in $\mathcal{AL}$ are PSpace-complete, and other results for $\mathcal{ALE}$ and $\mathcal{ALU}$.

A second possible restriction is to allow for axioms of the form $A \equiv C$, but in which a concept name can appear only *once* on the left-hand side. For axioms of this form in $\mathcal{ALN}$, Küsters [1998] proved that reasoning is PSpace-complete when the TBox is cyclic, and NP-complete when it is acyclic.

3.6 Undecidability

One of the main reasons why satisfiability and subsumption in many DLs are decidable—although highly complex—is that most of the concept constructors can express only *local* properties about an element [Vardi, 1997; Libkin, 2000]. Let C be a concept in $\mathcal{ALC}$: recalling the tableaux methods in Chapter 2, an assertion $C(x)$ states properties about x, and about elements which are linked to x by a chain of at most $|C|$ role assertions. Intuitively, this implies that a constraint regarding x will not "talk about" elements which are arbitrarily far (w.r.t. role links) from x. This also means that in $\mathcal{ALC}$, and in many DLs, an assertion on an individual cannot state properties about a whole structure satisfying it. However, not every DL satisfies locality.

3.6.1 Undecidability of role-value-maps

The first notable non-local DL is a subset of the language of the knowledge representation system Kl-One, isolated by Schmidt-Schauß [1989], which we call $\mathcal{FL}^{-}(\circ, =)$[1]. It contains conjunction, universal quantification, role composition, and *equality role-value-maps* $R = Q$. A role-value-map allows one to express concepts like "persons whose co-workers coincide with their relatives", as it could be, e.g., a small family-based firm. Using two role names co-worker and relative, this concept would be expressed as (co-worker = relative).

The DL proved undecidable by Schmidt-Schauß used equality role-value-maps.

[1] In his paper, Schmidt-Schauß used the name $\mathcal{ALR}$.

Table 3.2. *Syntax and semantics of the description logic $\mathcal{FL}^-(\circ,\subseteq)$.*

	concept expressions	semantics
concept name	A	$\subseteq \Delta^{\mathcal{I}}$
value restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y.\, (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
concept intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
role-value-map	$R \subseteq Q$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y.\, (x,y) \in R^{\mathcal{I}} \rightarrow (x,y) \in Q^{\mathcal{I}}\}$
	role expressions	**semantics**
role name	P	$\subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
role composition	$R \circ Q$	$\{(x,y) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \exists c.\, (x,z) \in R^{\mathcal{I}}, (z,y) \in Q^{\mathcal{I}}\}$

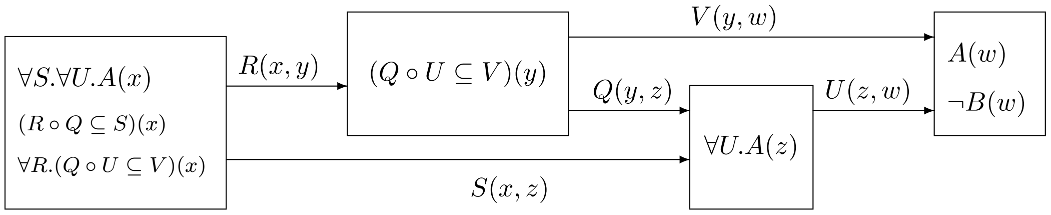
Here we present a simpler proof for a DL using *containment* role-value-maps $R \subseteq Q$. We call this DL $\mathcal{FL}^-(\circ,\subseteq)$. Clearly, $\mathcal{FL}^-(\circ,\subseteq)$ is (slightly) more expressive than $\mathcal{FL}^-(\circ,=)$, since $R = Q$ can be expressed by $(R \subseteq Q) \sqcap (Q \subseteq R)$, but not vice versa. Most of the original reduction is preserved, though.

Although all constructs of $\mathcal{FL}^-(\circ,\subseteq)$ have already been defined in different parts of Chapter 2, we recall for convenience their syntax and semantics in the single Table 3.2. Recall that $R \subseteq Q$ is a concept; namely, the concept of all elements whose set of fillers for role R is included in the set of fillers for role Q. To avoid many parentheses, we assume $\circ$ has always precedence over $\subseteq$.

Before giving the proof that subsumption in $\mathcal{FL}^-(\circ,\subseteq)$ is undecidable, let us consider an example illustrating why $\mathcal{FL}^-(\circ,\subseteq)$ is not local.

Example 3.29 Let Q, R, S, U, V be role names. Consider whether the concept $C = \forall S.\forall U.A \sqcap (R \circ Q \subseteq S) \sqcap \forall R.(Q \circ U \subseteq V)$ is subsumed by the concept $D = \forall R.\forall Q.\forall U.B$.

The answer is no: in fact, a model satisfying C and not satisfying D is shown in Fig. 3.2. This model can be obtained trying to satisfy $\neg D = \exists R.\exists Q.\exists U.\neg B$ with individual x, y, z, w, and then adding role assertions satisfying C. Observe that a model of C cannot be a tree because of concepts like $(R \circ Q \subseteq S)$. Hence, any notion of "distance" between two individuals in a model, as number of role links connecting them, is ambiguous when a DL has role-value-maps. Moreover, the satisfaction of the assertions $(R \circ Q \subseteq S)(x)$ and $\forall S.A(x)$ in an interpretation depends on the satisfaction of the assertion $A(z)$, for every individual z connected to x via a path of role fillers that can be composed according to role-value-maps. In fact, replacing B with A in D yields a concept D' which now subsumes C—and indeed, the previous model satisfies also D'. ∎



$\forall S.\forall U.A(x)$
$(R \circ Q \subseteq S)(x)$
$\forall R.(Q \circ U \subseteq V)(x)$
$R(x,y)$
$(Q \circ U \subseteq V)(y)$
$V(y,w)$
$Q(y,z)$
$\forall U.A(z)$
$U(z,w)$
$A(w)$
$\neg B(w)$
$S(x,z)$

Fig. 3.2. A possible countermodel for $C \sqsubseteq D$ in Example 3.29. Boxes group assertions about an individual; arrows represent role assertions.

These properties are crucial for the reduction from ground rewriting systems to subsumption in $\mathcal{FL}^-(\circ, \subseteq)$. For basics about rewriting systems, consult [Dershowitz and Jouannaud, 1990].

Definition 3.30 (Ground Rewriting System) Let Σ be a finite alphabet $\{a, b, \ldots\}$. A *term* w on Σ is an element of Σ^*, i.e., a finite sequence of 0 or more letters from Σ. If v, w are terms, their *concatenation* is a term, denoted as vw. A ground *rewriting system* is a finite set of rewriting rules $\rho = \{s_i \rightarrow t_i\}_{i=1,\ldots,n}$, where for every $i \in \{1, \ldots, n\}$ both s_i and t_i are terms on Σ. The *rewriting relation* $\stackrel{*}{\rightarrow}$ induced by a set of rewriting rules ρ is the minimal relation which is reflexive, transitive, and satisfies the following conditions:

(i) if $s \rightarrow t \in \rho$ then $s \stackrel{*}{\rightarrow} t$;
(ii) for every letter $a \in \Sigma$, if $p \stackrel{*}{\rightarrow} q$ then both $ap \stackrel{*}{\rightarrow} aq$ and $pa \stackrel{*}{\rightarrow} qa$.

The *rewriting problem* for ground rewriting systems is: Given a set of rewriting rules ρ and two terms v, w, decide whether $v \stackrel{*}{\rightarrow} w$. ■

Remark 3.31 In general, a single rewriting step of a term v consists in finding a substring of v which coincides with the antecedent s of a rewriting rule $s \rightarrow t$, and then substitute t for s in v. Hence, $v \stackrel{*}{\rightarrow} w$ if there exist n terms $u_1, \ldots, u_n$ such that $u_1 = v, u_n = w$, and for each $i \in 1..n-1$ the two terms u_i, u_{i+1} are such that for some terms p and q, it is $u_i = psq$, $u_{i+1} = ptq$, and $s \rightarrow t \in \rho$. This proves that the term problem is recursively enumerable. However, it is semidecidable (recursively enumerable, but nonrecursive). ■

We reduce this problem to subsumption in $\mathcal{FL}^-(\circ, \subseteq)$ as follows. First of all, observe that we can define the following one-to-one correspondence between terms and role chains:

- for every letter a in Σ, let P_a be a role name;
- for every term w, let R_w be the composition of the role names corresponding to the letters of w. For example, if $w = aab$, then $R_w = P_a \circ P_a \circ P_b$.

Now for each set of rewriting rules ρ, we define the concept C_ρ as

$$C_\rho = \sqcap_{s \to t \in \rho}(R_s \subseteq R_t)$$

Let Q be a new atomic role: we define a concept C_Σ as

$$C_\Sigma = \sqcap_{a \in \Sigma}(Q \circ P_a \subseteq Q)$$

Intuitively, if a model $\mathcal{I}$ satisfies $C_\Sigma(x)$, then for every term w, if $(Q \circ R_w)(x, z)$ holds in $\mathcal{I}$, then $Q(x, z)$ also holds, i.e., x is directly connected via Q to every other element z to which it is indirectly connected via $Q \circ R_w$.

If also $\mathcal{I} \models \forall Q.C_\rho(x)$, then $C_\rho(z)$ holds for every such z. This is a key property of the reduction.

Remark 3.32 The two concepts $\forall Q.C_\rho$ and C_Σ are a way to internalize simple axioms in a concept. Consider a TBox $\mathcal{T} = \{\top \sqsubseteq C_\rho\}$ which states that every individual in a model must satisfy concept C_ρ. One could prove that in $\mathcal{FL}^-(\circ, \subseteq)$ a concept C is subsumed by a concept D w.r.t. $\mathcal{T}$ iff $C_\Sigma \sqcap \forall Q.C_\rho \sqcap \forall Q.C$ is subsumed by $\forall Q.D$, where the latter is plain subsumption between concept expressions. ∎

Theorem 3.33 *Subsumption in $\mathcal{FL}^-(\circ, \subseteq)$ is undecidable.*

Let ρ be a set of rewriting rules, and v, w be two terms. Define the following two concepts:

$$C = C_\Sigma \sqcap \forall Q.C_\rho \tag{3.17}$$

$$D = \forall Q.(R_v \subseteq R_w) \tag{3.18}$$

We divide the proof in two lemmata.

Lemma 3.34 *If $v \xrightarrow{*} w$ then the concept C is subsumed by D.*

Proof We first prove that the claim holds for the base case of the inductive definition of $\xrightarrow{*}$ (Condition (i) in Definition 3.30). Then, we prove the claim for the two inductive cases (Condition (ii)). Finally, we prove that the proof carries over the closure conditions. In all cases, let $s \to t \in \rho$.

Base case. The concept D is $\forall Q.(R_s \subseteq R_t)$. Observe that the concept $\forall Q.C_\rho$ is equivalent to $\sqcap_{s \to t \in \rho} \forall Q.(R_s \subseteq R_t)$. Hence, C is subsumed by D because D is one of the conjuncts of (an equivalent form of) C.

Inductive cases. For the first inductive case, let $D = \forall Q.(P_a \circ R_p \subseteq P_a \circ R_q)$, and let the inductive hypothesis be that C is subsumed by $\forall Q.R_p \subseteq R_q$. By refutation, suppose C is not subsumed by D: then, there is a model $\mathcal{I}$ in which both $C(x)$ and $\neg D(x)$ hold. The latter constraint implies that there is an element y such that

(i) $\mathcal{I} \models Q(x, y)$
(ii) $\mathcal{I} \models (P_a \circ R_p)(y, z)$
(iii) $\mathcal{I} \not\models (P_a \circ R_q)(y, z)$

From (ii), there is an element y' such that both $P_a(y, y')$ and $R_s(y', z)$ hold. Now from $C_\Sigma(x)$, it must be $\mathcal{I} \models Q(x, y')$, and from the inductive hypothesis this implies $(R_s \subseteq R_t)(y')$. Then, $\mathcal{I} \models R_t(y', z)$ holds, hence $\mathcal{I} \models (P_a \circ R_t)(y, z)$, contradicting (iii).

The second inductive case is simpler, since one does not need to consider $C_\Sigma(x)$. The interested reader can use it as an exercise.

We conclude the proof by showing that the reduction carries over the reflexive and transitive closure of $\overset{*}{\rightarrow}$.

First, from the semantics in Table 3.2 follows that $R_w \subseteq R_w$ is equivalent to $\top$, which implies also that $D \equiv \top$. Hence the claim holds also for $w \overset{*}{\rightarrow} w$ (i.e., reflexivity).

For transitivity, the induction is easy: suppose $u \overset{*}{\rightarrow} v$ and $v \overset{*}{\rightarrow} w$: then by induction C is subsumed by D_1 and by D_2, where $D_1 = \forall Q.(R_u \subseteq R_v)$ and $D_2 = \forall Q.(R_v \subseteq R_w)$. Then C is subsumed also by $D_1 \sqcap D_2$ which is equivalent to $\forall Q.((R_u \subseteq R_v) \sqcap (R_v \subseteq R_w))$. This concept is subsumed by $\forall Q.(R_u \subseteq R_w)$, which is the claim. □

We now prove the other direction of the reduction.

Lemma 3.35 *If $v \overset{*}{\not\rightarrow} w$, then the concept C is not subsumed by D.*

Proof We give the rule to construct an infinite tableau branch $\mathcal{T}$ and show that it defines a model that satisfies C, and does not satisfy D. The tableau is one-one with an infinite automaton accepting the term v, and every other term v can be rewritten into. Let $v[1], \ldots, v[n]$ denote the n letters of v ($v[i]$ is the i-th letter of v).

Let x, y, z be individual names. Start from the set of assertions

$$\mathcal{T}_0 = P_{v[1]}(y, y_1), \ldots, P_{v[i+1]}(y_i, y_{i+1}), \ldots, P_{v[n]}(y_{n-1}, z)$$

Then add role assertions to $\mathcal{T}$ following the $\rightarrow_\subseteq$**-rule**:

Condition: there is a rewriting rule $s \rightarrow t \in \rho$
where $s = s[1] \cdots s[h]$ and $t = t[1] \cdots t[k]$;
$\mathcal{T}$ contains $h + 1$ individuals $y_0, \ldots, y_h$ and h assertions
$P_{s[i]}(y_{i-1}, y_i)$ for $i \in \{1, \ldots, h\}$

$\mathcal{T}$ does not contain all assertions $P_{t[1]}(y_0, y'_1), \ldots, P_{t[n]}(y'_{k-1}, y_h)$
Action: $\mathcal{T}' = \mathcal{T} \cup \{P_{t[1]}(y_0, y'_1), \ldots, P_{t[n]}(y'_{k-1}, y_h)\}$,
where $y'_1, \ldots, y'_{k-1}$, are $k-1$ individual names not occurring in $\mathcal{T}$.

Intuitively, if there is in $\mathcal{T}$ a path of role assertions such that $R_s(y_0, y_h)$ holds, the $\rightarrow_{\subseteq}$-rule adds another path such that also $R_t(y_0, y_h)$ holds. Of course, $\mathcal{T}_\omega$ can have an infinite number of individuals and role assertions between them; this is reasonable, since its role paths from y to z are one-one with the possible transformations on v one can make using the rewriting rules. One can also think $\mathcal{T}_\omega$ as an infinite-state automaton accepting $\overline{v} = \{u \mid v \overset{*}{\rightarrow} u\}$.

The $\rightarrow_{\subseteq}$-rule always adds new assertions to $\mathcal{T}$, and its application given some premises does not destroy other premises of application of the $\rightarrow_{\subseteq}$-rule itself, since we keep in $\mathcal{T}$ all the rewritten terms. Therefore, the construction is monotonic over the $\subseteq$-lattice of all tableaux with a countable number of individuals, and role assertions between individuals. In building $\mathcal{T}_\omega$, however, a *fair strategy* must be adopted. That is, if at a given stage $\mathcal{T}_i$ of the construction, the $\rightarrow_{\subseteq}$-rule is applicable for individuals $y_0, \ldots, y_h$, then for some finite k, in $\mathcal{T}_{i+k}$ the $\rightarrow_{\subseteq}$-rule has been applied for those premises—i.e., a possible rule application is not indefinitely deferred. This could be achieved by, e.g., inserting possible rule applications in a queue.

Proposition 3.36 *Let $\mathcal{T}_\omega$ be constructed using the $\rightarrow_{\subseteq}$-rule, and a fair strategy. For every term $u = u[1] \cdots u[k]$, $v \overset{*}{\rightarrow} u$ iff in $\mathcal{T}_\omega$ there are $k-1$ individual names $y_1, \ldots, y_{k-1}$ and k assertions $P_{u[1]}(y, y_1), \ldots P_{u[k]}(y_{k-1}, z)$.*

Proof If $v \overset{*}{\rightarrow} u$, then there are a minimum finite number n of applications of rewriting rules in ρ transforming v into u. By induction on such n, the premises of the $\rightarrow_{\subseteq}$-rule are fulfilled, and since $\mathcal{T}_\omega$ is built adopting a fair strategy, from some finite stage of its construction onwards, $R_u(y, z)$ must hold. For the other direction, if $R_u(y, z)$ holds in $\mathcal{T}_\omega$, then for each $\rightarrow_{\subseteq}$-rule application leading to $R_u(y, z)$ one can apply a rewriting rule to v, leading to u. □

We can now define the model $\mathcal{I}$ satisfying C and not satisfying D. Let N be the set of individual names of $\mathcal{T}_\omega$. $\mathcal{I}$ has domain $\{x\} \cup N$. Let $\mathcal{I} = \mathcal{T}_\omega \cup \{Q(x, y) | y \in N\}$. Then $\mathcal{I}$ satisfies $C(x)$ straightforwardly; moreover, it does not satisfy D from Proposition 3.36. □

To prove that subsumption in undecidable in the less expressive DL $\mathcal{FL}^-(\circ, =)$, Schmidt-Schauß [1989] started from the word problem for groups. Starting from the Post correspondence problem, with a more complex construction, also Patel-Schneider [1989b] proved that subsumption is undecidable in the more expressive DL $\mathcal{FL}^-(\circ, \subseteq)$ plus role inverses, functional roles, and role restrictions.

Starting from the *word problem*—which is less general than the term rewriting problem, but still semidecidable—Baader [1998] showed that subsumption in $\mathcal{FL}^-(\circ, \subseteq)$ is undecidable without referring to tableaux. We report here the second part of his proof, (corresponding to Lemma 3.35) since it is quite short and elegant, and shows a different way of proving the only-if direction, namely, giving a direct definition of an infinite structure satisfying the concepts.

The word problem follows Definition 3.30, but considers the reflexive-symmetric-transitive closure $\overset{*}{\leftrightarrow}$ of rewriting rules. This is also known as the word problem for semigroups, or Thue systems. In this case, *ground term* and *word* are synonyms. Of course, $\overset{*}{\leftrightarrow}$ is an equivalence relation on words; let $[v]$ denote the $\overset{*}{\leftrightarrow}$-equivalence classes. Note that $[u] = [v]$ iff $u \overset{*}{\leftrightarrow} v$. There is a natural multiplication on these classes induced by concatenation: $[u][v] = [uv]$ (since $\overset{*}{\leftrightarrow}$ is even a congruence, this is well-defined).

Taking the equivalence classes plus one distinguished element x as domain of the model $\mathcal{I}$, the roles can be interpreted as

$$Q^{\mathcal{I}} = \{(x, [u]) | u \in \Sigma^*\} \tag{3.19}$$

$$(P_a)^{\mathcal{I}} = \{([u], [ua]) | a \in \Sigma, u \in \Sigma^*\} \tag{3.20}$$

Then, it can be shown that if $v \overset{*}{\not\leftrightarrow} w$, then x belongs to $C^{\mathcal{I}}$, but not to $D^{\mathcal{I}}$ as follows.

(i) x belongs to $C^{\mathcal{I}}$: from (3.20), for every word u it is $(x, [u]) \in Q^{\mathcal{I}}$ and $([u], [ua]) \in (P_a)^{\mathcal{I}}$; but also from (3.19), $(x, [ua]) \in Q^{\mathcal{I}}$, hence $C_\Sigma(x)$ is satisfied by $\mathcal{I}$. Regarding $\forall Q.C_\rho(x)$, suppose $([u], [w]) \in (R_s)^{\mathcal{I}}$, where $s \to t \in \rho$. Then $[w] = [us]$ by definition of $(P_a)^{\mathcal{I}}$. Moreover, from $s \to t \in \rho$ it follows $us \overset{*}{\leftrightarrow} ut$, hence $[us] = [ut]$. Consequently, $([u], [w]) = ([u], [ut]) \in (R_t)^{\mathcal{I}}$ from (3.20).

(ii) x does not belong to $D^{\mathcal{I}}$: for the empty word ϵ, $[\epsilon]$ is a Q-filler of x, however $[\epsilon]$ does not satisfy the concept $R_v \subseteq R_w$. In fact, $([\epsilon], [v]) \in (R_v)^{\mathcal{I}}$, but not $([\epsilon], [v]) \in (R_w)^{\mathcal{I}}$ since $[w]$ is the only R_w-filler of $[\epsilon]$, but $[v] \neq [w]$ from the assumption that $v \overset{*}{\not\leftrightarrow} w$.

3.7 Reasoning about individuals in ABoxes

When an ABox is considered, the reasoning problem of *instance check* arises: Given an ABox $\mathcal{A}$, an individual a and a concept C, decide whether $\mathcal{A} \models C(a)$. For the instance check problem, the size of the input is formed by the size of the concept expression C plus the size of $\mathcal{A}$. Since the size of one input may be much larger than the other in real applications, it makes sense to distinguish the complexity w.r.t. the two inputs—as it is usually done in databases with data complexity and query complexity [Vardi, 1982].

A common intuition [Schmolze and Lipkis, 1983] about instance check was that it could be performed via subsumption, using the so-called most specific concept (msc) method.

Definition 3.37 (most specific concepts) Let $\mathcal{A}$ be an ABox in a given DL, and let a be an individual in $\mathcal{A}$. A concept C is the *most specific concept* of a in $\mathcal{A}$, written $msc(\mathcal{A}, a)$, if, for every concept D in the given DL, $\mathcal{A} \models D(a)$ implies $C \sqsubseteq D$. ■

Recall from Chapter 2 a slightly different definition of msc in the *realization problem*: given an individual a and an ABox $\mathcal{A}$, find the most specific concepts C (w.r.t. subsumption) such that $\mathcal{A} \models C(a)$ [Nebel, 1990a, p. 104]. Since conjunction is always available in every DL, the two definitions are equivalent (just conjoin all specific concepts of realization in one msc).

Clearly, once $msc(\mathcal{A}, a)$ is known, to decide whether a is an instance of a concept D it should be sufficient to check whether $msc(\mathcal{A}, a)$ is subsumed by D, turning instance checking into subsumption. Moreover, when a TBox is present, off-line classification of all msc's in the TBox may provide a way to pre-compute many instance checks, providing an on-line speed-up.

The intuition about how computing $msc(\mathcal{A}, a)$ was to gather the concepts/properties explicitly stated for a in $\mathcal{A}$. However, this approach is quite sensitive to the DL chosen to express $msc(\mathcal{A}, a)$ and the queries. In fact, most specific concepts can be easily computed for simple DLs, like $\mathcal{AL}$. However, it may not be possible when slightly more expressive languages are considered.

Example 3.38 A simple example (simplified from [Baader and Küsters, 1998]) is the ABox made just by the assertion $R(a, a)$. If $\mathcal{FL}^-$ is used for most specific concepts and queries, then $msc(\{R(a, a)\}, a) = \exists R$. However, if qualified existential quantification is allowed for most specific concepts, then each of the concepts $\exists R, \quad \exists R.\exists R, \quad \exists R.\exists R.\exists R, \ \ldots,$ is more specific than the previous one. Using this argument, it is possible to prove that $msc(\{R(a, a)\}, a)$ has no finite representation, unless also transitive closure on roles is allowed. Using the axiom $A \sqsubseteq \exists R.A$ in an ad-hoc TBox, $msc(\{R(a, a)\}, a) = A$ for the simple ABox of this example—but this does not simplify instance check. An alternative approach would be to raise individuals in the language to express concepts, through the concept constructor $\{\ldots\}$ that enumerates the individuals belonging to it (called "one-of" in CLASSIC). In that case, $msc(\{R(a, a)\}, a) = \exists R.\{a\}$ (see [Donini *et al.*, 1990]). But this "solution" to instance check becomes now a problem for subsumption, which must take individuals into account (for a treatment of DLs with one-of, see [Schaerf, 1994a]). ■

The msc's method makes an implicit assumption: to work well, the size of $msc(\mathcal{A}, a)$ should be comparable with the size of the whole ABox, and in most cases much shorter. However, consider the DL $\mathcal{ALE}$, in which subsumption is in NP. Then, solving instance check by means of subsumption in polynomial space and time would imply that in instance check was in NP, too. However, suppose that we prove that instance check was hard for coNP. Then, solving instance check by subsumption implies that either *co*NP $\subseteq$ NP, or $msc(\mathcal{A}, a)$, if ever exists, has superpolynomial size w.r.t. $\mathcal{A}$. The former conclusion is unlikely to hold, while the latter would make unfeasible the entire method of msc's.

In general, this argument works whenever subsumption in a DL belongs to a complexity class $\mathcal{C}$, while instance check is proved hard for a different complexity class $\mathcal{C}'$, for which $\mathcal{C}' \subseteq \mathcal{C}$ is believed to be false. We present here a proof using this argument, found by Schaerf [1993; 1994b; 1994a].

We first start with a simple example highlighting the construction.

Example 3.39 Let f, c_1, c_2, x, y, z be individuals, R, P, N be role names, and A a concept name. Let $\mathcal{A}$ be the following ABox, whose structure we highlight using some arrows between assertions:

$$\begin{tikzcd}
 & & P(c_1,x) & A(x) \\
 & R(f,c_1) \arrow[ur] \arrow[dr] & & \\
f \arrow[ur] \arrow[ddr] & & N(c_1,y) & \\
 & & P(c_2,y) & \\
 & R(f,c_2) \arrow[ur] \arrow[dr] & & \\
 & & N(c_2,z) & \neg A(z)
\end{tikzcd}$$

The query $\exists R.(\exists P.A \sqcap \exists N.\neg A)(f)$ is entailed by $\mathcal{A}$. That is, one among c_1 and c_2 has its P-filler in A and its N-filler in $\neg A$. This can be verified by *case analysis* on y: in every model either $A(y)$ or $\neg A(y)$ must be true. For models in which $A(y)$ holds, c_2 is the R-filler of f satisfying the query; for models in which $\neg A(y)$ holds, c_1 is. Observe that if $\mathcal{ALE}$ is used to express most specific concepts, the best approximation we can find for $msc(\mathcal{A}, f)$, by collecting assertions along the role paths starting from f, is the concept $C = \exists R.(\exists P.A \sqcap \exists N) \sqcap \exists R.(\exists P \sqcap \exists N.\neg A)$, in which the fact that the *same* individual y is both the N-filler of $\exists N$ and the P-filler of $\exists P$ is lost. Indeed, C is *not* subsumed by the query, as one can see constructing an open tableau for $C \sqcap \neg\exists R.(\exists P.A \sqcap \exists N.\neg A)(f)$. ■

The above example can be extended to a proof that deciding $\mathcal{A} \models C(a)$, where C is an $\mathcal{ALE}$-concept, is coNP-hard. Observe that this is a different source of complexity w.r.t. unsatisfiability in $\mathcal{ALE}$. In fact, a concept C is unsatisfiable iff $\{C(a)\} \models \bot(a)$. This problem is NP-complete when C is a concept in $\mathcal{ALE}$ (Section 3.3.1).

The source coNP-complete problem is the complement of 2+2-SAT, which is the following problem.

Definition 3.40 (2+2-sat) Given a 4CNF propositional formula F, in which every clause has exactly two positive literals and two negative ones, decide whether F is satisfiable. ■

The problem 2+2-SAT is a simple variant of the well-known 3-SAT. Indeed, for 3-literal clauses mixing both positive and negative literals, add a fourth disjunct, constantly false; e.g., $X \vee Y \vee \neg Z$ is transformed into the 2+2-clause $X \vee Y \vee \neg Z \vee \neg\mathbf{true}$. Unmixed clauses can be replaced by two mixed ones using a new variable (see [Schaerf, 1994a, Theorem 4.2.6]).

Given an instance of 2+2-SAT $F = C_1 \wedge C_2 \wedge \cdots \wedge C_n$, where each clause $C_i = L^i_{1+} \vee L^i_{2+} \vee \neg L^i_{1-} \vee \neg L^i_{2-}$, we construct an ABox $\mathcal{A}_F$ as follows. $\mathcal{A}_F$ has one individual l for each variable L in F, one individual c_i for each clause C_i, one individual f for the whole formula F, plus two individuals $true$ and $false$ for the corresponding propositional constants.

The roles of $\mathcal{A}_F$ are Cl (for Clause), P_1, P_2 (for positive literals), N_1, N_2 (for negative literals), and the only concept name is A. Finally, $\mathcal{A}_F$ is given by (we group role assertions on first individual to ease reading):

$$\begin{array}{lll}
Cl(f, c_1) & \left\{ \begin{array}{l} P_1(c_1, l^1_{1+}) \\ P_2(c_1, l^1_{2+}) \\ N_1(c_1, l^1_{1-}) \\ N_2(c_1, l^1_{2-}) \end{array} \right. & \\
\vdots & \quad \vdots & A(true), \quad \neg A(false) \\
Cl(f, c_n) & \left\{ \begin{array}{l} P_1(c_n, l^n_{1+}) \\ P_2(c_n, l^n_{2+}) \\ N_1(c_n, l^n_{1-}) \\ N_2(c_n, l^n_{2-}) \end{array} \right. &
\end{array}$$

Now let D be the following, fixed, query concept:

$$D = \exists Cl.((\exists P_1.\neg A) \sqcap (\exists P_2.\neg A) \sqcap (\exists N_1.A) \sqcap (\exists N_2.A))$$

Intuitively, an individual name l is in the extension of A or $\neg A$ iff the propositional variable L is assigned **true** or **false**, respectively. Then, checking whether $\mathcal{A}_F \models D(f)$ corresponds to checking that in every truth assignment for F there exists a clause whose positive literals are interpreted as false, and whose negative literals are interpreted as true—i.e., a clause that is not satisfied. If one applies the above idea to translate the two clauses (having just two literals each one) $\mathbf{false} \vee \neg Y$, $Y \vee \neg\mathbf{true}$, one obtains exactly the ABox of Example 3.39.

The correctness of this reduction was proved by Schaerf [1993; 1994a]. We report here only the concluding lemma.

Lemma 3.41 *A 2+2-CNF formula F is unsatisfiable if and only if $\mathcal{A}_F \models D(f)$.*

Hence, instance checking in $\mathcal{ALE}$ is coNP-hard. This implies that instance check in $\mathcal{ALE}$ cannot be efficiently solved by subsumption, unless *co*NP $\subseteq$ NP. We remark that only the size of $\mathcal{A}_F$ depends on the source formula F, while D is fixed. Hence, instance checking in $\mathcal{ALE}$ is coNP-hard with respect to *knowledge base* complexity—and it is also NP-hard from Section 3.3.1. The upper bound for knowledge base complexity of instance checking in $\mathcal{ALE}$ is in Π_2^p, but it is still not known whether the problem is Π_2^p-complete. Regarding combined complexity—that is, neither the size of the ABox nor that of the query is fixed—in [Schaerf, 1994a; Donini *et al.*, 1994b] it was proved that instance checking in $\mathcal{ALE}$ is PSPACE-complete.

Since the above reduction makes use of negated concept names, it may seem that coNP-hardness arises from the interaction between qualified existential quantification and negated concept names. However, all it is needed are two concepts whose union covers all possible cases. We saw in Section 3.2.1 that also $\exists R$ and $\forall R.B$ have this property. Therefore, if we replace A and $\neg A$ in $\mathcal{A}_F$ with $\exists R$ and $\forall R.B$, respectively, (where R is a *new* role name and B is a *new* concept name), we obtain a new reduction for which Lemma 3.41 still holds. Hence, instance checking in $\mathcal{FL}^-\mathcal{E}$ (i.e., $\mathcal{ALE}$ without negation of concept names) is coNP-hard too, thus confirming that coNP-hardness is originated by qualified existential quantification alone. In other words, intractability arises from a query language containing both qualified existential quantification, and pairs of concepts whose union is equivalent to $\top$. Hence, for languages containing these constructs, the msc method is not effective.

Regarding the expressivity of the language for assertions in the ABox, coNP-hardness of instance checking arises already when assertions in the ABox involve just concept and role names. However, note that a key point in the reduction is the fact that two individuals in the ABox can be linked via different role paths, as f and y were in Example 3.41.

3.8 Discussion

In this chapter we analyzed various lower bounds on the complexity of reasoning about simple concept expressions in DLs. Our presentation appealed to the intuitive notions of exploring AND-OR trees, in the special case when the tree comes out of a tableau.

We remark that an alternative approach to reasoning is to reduce it to the emptiness test for automata (e.g., [Vardi, 1996]), which has been quite successfully applied to temporal logics, and propositional logics of programs. However, till now such techniques were used to obtain upper bounds in reasoning, while in order obtaining lower bounds one would need a way to reduce problems on automata to unsatisfiability/subsumption in DL. The only example of this reduction is [Nebel, 1990b], for a very simple DL, which we did not presented in this chapter for lack of space.

We end the chapter with a perspective on the significance of the NP, coNP, and PSpace complexity lower bounds we presented. Present reasoning systems in DLs (see chapter in this book) can now cope with reasonable size ExpTime-complete problems. Hence the computational complexity of the problems now reachable is above PSpace. However, in our opinion, for implemented systems the significance of a reduction lies not just in the theoretical lower bound obtained, but also in the reduction itself. In fact, when experimenting algorithms for subsumption, satisfiability, etc. [Baader *et al.*, 1992a; Hustadt and Schmidt, 1997] on an implemented system, one can exploit already known "hard" cases of a source problem like 3-SAT, 2+2-SAT, SET SPLITTING, or QBF validity to obtain "hard" instances for the algorithm under test. These instances isolate the influence of each source of combinatorial explosion on the performance of the overall reasoning system, and can be used to optimize reasoning algorithms in a piecewise fashion [Horrocks and Patel-Schneider, 1999], separately for the various sources of complexity. In this respect, the issue of finding "efficient" reductions (w.r.t. the size of the resulting concepts) is still open, and can make the difference when concepts to be tested scale up (see [Donini and Massacci, 2000]).

3.9 A list of complexity results for subsumption and satisfiability

A lot of names were invented for languages of different DLs, e.g., $\mathcal{FL}$ for Frame Language, $\mathcal{ALC}$ for Attributive Descriptions Language with Complement, etc. Although suggestive, these names are not very explicit about which constructs are in the named language. This makes the huge mass of results about complexity of reasoning in DLs often difficult to screen by non-experts in the field. To clarify the constructs each language is equipped with, we use two lists of constructors: the first one for concept constructors, and the second one for role constructors. For example, the pair of lists $(\sqcap, \exists R, \forall R.C)$ $(\sqcap, \circ)$ denotes a language whose concept constructors are conjunction $\sqcap$, unqualified existential quantification $\exists R$, universal role quantification $\forall R.C$, and whose role constructors are conjunction $\sqcap$ and composition $\circ$. Many combinations of concept constructors have been given a name which is now commonly used. For instance, the first list of the above example is known as $\mathcal{FL}^-$. In these cases, we follow a syntax first proposed in [Baader and Sattler, 1996b], and write just $\mathcal{FL}^-(\sqcap, \circ)$—that is, $\mathcal{FL}^-$ augmented with role conjunction and composition—to make it immediately recognizable also by researchers in the field.

3.9.1 Notation

In the following catalog, satisfiability and subsumption refer to the problems with plain concept expressions. When satisfiability and subsumption are w.r.t. a set of axioms, we state it explicitly. Moreover, when the constructs of the DL allows one to reduce subsumption between C and D to satisfiability of $C \sqcap \neg D$, we mention only satisfiability.

In the lists, we tried to use the symbol of the DL construct whenever possible. We abbreviated some constructs, however: unqualified number restrictions $\geqslant n\, R$, $\leqslant n\, R$ are denoted as $\lesseqgtr R$, while qualified number restrictions $\geqslant n\, R.C$, $\leqslant n\, R.C$ are $\lesseqgtr R.C$. When a construct is allowed only for names (either concept names in the first list, or role names in the second one) we apply the construct to the word *name*.

3.9.2 Subsumption in PTime

To the best of author's knowledge, no proof of PTime-hardness was given for any DL so far. Therefore the following results refer only to membership in PTime.

- $(\sqcap, \exists R, \forall R.C)$ () known as $\mathcal{FL}^-$ [Levesque and Brachman, 1987].
- $(\sqcap, \exists R, \forall R.C, \neg(\mathit{name}))$ () known as $\mathcal{AL}$ [Schmidt-Schauß and Smolka, 1991]
- $(\sqcap, \exists R, \forall R.C, \lesseqgtr R)$ () known as $\mathcal{ALN}$ [Donini *et al.*, 1997a]
- $\mathcal{AL}(\circ), \mathcal{AL}(^-)$ [Donini *et al.*, 1999]
- $\mathcal{FL}^-(\sqcap)$ [Donini *et al.*, 1991a]
- $(\sqcap, \exists R.C, \{\mathit{individual}\})$ $(\sqcap, ^-)$ known as $\mathcal{ELIRO}^1$ [Baader *et al.*, 1998b]

3.9.3 NP and coNP

- $(\sqcap, \exists R.C, \forall R.C, \neg(\mathit{name}))$ () (known as $\mathcal{ALE}$) subsumption and unsatisfiability are NP-complete [Donini *et al.*, 1992a] (see Section 3.3.1)
- $\mathcal{AL}(\sqcap)$, $\mathcal{ALE}(\sqcap)$, and $(\sqcap, \exists R.C, \forall R.C)$ () (known as $\mathcal{ALR}$, $\mathcal{ALER}$ and $\mathcal{FL}^-\mathcal{E}$ respectively) subsumption and unsatisfiability NP-complete [Donini *et al.*, 1997a] (see Theorems 3.16,3.17 for hardness, and [Donini *et al.*, 1992a] for membership)
- $(\sqcap, \sqcup, \exists R, \forall R.C, \neg(\mathit{name}))$ () (known as $\mathcal{ALU}$) subsumption and unsatisfiability coNP-complete [Donini *et al.*, 1997a] (see Section 3.1.1.1)
- $\mathcal{ALN}(^-)$ subsumption is coNP-complete, while satisfiability is decidable in polynomial time [Donini *et al.*, 1999]

- $\mathcal{FL}^-(\sqcap,^-)$, $\mathcal{FL}^-(\sqcap,\circ)$, and $\mathcal{FL}^-(\circ,^-)$ [Donini *et al.*, 1999] (see Sections 3.3.2,3.3.3, and 3.3.4)
- $\mathcal{AL}()$, satisfiability w.r.t. a set of *acyclic* axioms is coNP-hard [Buchheit *et al.*, 1994a; Calvanese, 1996b; Buchheit *et al.*, 1998] (coNP-complete for $\mathcal{ALE}()$ [Calvanese, 1996b]).

3.9.4 PSpace

- $(\sqcap,\sqcup,\neg,\exists R.C,\forall R.C)$ () (known as $\mathcal{ALC}$) [Schmidt-Schauß and Smolka, 1991] (see Section 3.4.1)
- $(\sqcap,\neg(name),\exists R.C,\forall R.C,\lessgtr R)$ () (known as $\mathcal{ALEN}$) [Hemaspaandra, 1999]
- $\mathcal{FL}^-(R|_C)$ (known as $\mathcal{FL}$), $\mathcal{ALN}(\sqcap)$, $\mathcal{ALU}(\sqcap)$, $(\sqcap,\exists R.C,\forall R.C,\neg,\lessgtr R)$ $(\sqcap)$ (known as $\mathcal{ALCNR}$) [Donini *et al.*, 1997a]
- $\mathcal{ALC}(\sqcap,\sqcup,\circ)$ satisfiability [Massacci, 2001]. Membership is nontrivial.
- $\mathcal{ALE}()$ satisfiability w.r.t. a set of cyclic axioms is PSpace-complete [Calvanese, 1996b].
- $\mathcal{ALN}()$ satisfiability w.r.t. a set of cyclic axioms of the form $A \equiv C$, where each concept name A can appear only once on the left-hand side, is PSpace-complete [Küsters, 1998].

3.9.5 ExpTime

- $\mathcal{AL}$ w.r.t. a set of axioms (see Section 3.5 for hardness).
- $(\sqcap,\sqcup,\neg,\exists R.C,\forall R.C)$ $(\sqcup,\circ,^*,id(),^-)$ which includes $\mathcal{ALC}_{trans}$ [Baader, 1991; Schild, 1991]. Membership is nontrivial, and was proved by Pratt [1979] without inverse, and by Vardi and Wolper [1986] for *converse*-PDL reducing the problem to emptiness of tree-automata.
- $(\sqcap,\sqcup,\neg,\exists R.C,\forall R.C,\lessgtr\ name.C,\lessgtr\ name^-.C)$ $(\sqcup,\circ,^*,^-,id())$, known as $\mathcal{ALCQI}_{reg}$ (see Chapter 5). Membership is nontrivial.
- $(\sqcap,\sqcup,\neg,\exists R.C,\forall R.C,\mu x.C[x],\{individual\})$ $(^-)$, where $\mu x.C[x]$ denotes the least fixpoint of x [Sattler and Vardi, 2001]. Membership is nontrivial.

3.9.6 NExpTime

- adding concrete domains (see [Baader and Hanschke, 1991a]), satisfiability in $\mathcal{ALC}$ w.r.t. a set of acyclic axioms, and $\mathcal{ALC}(^-)$ [Lutz, 2001a]
- $\mathcal{ALC}(\sqcap,\sqcup,\neg)$ satisfiability [Lutz and Sattler, 2001]
- $(\sqcap,\sqcup,\exists R.C,\forall R.C,\neg,\{individual\},\lessgtr R.C)$ () satisfiability [Tobies, 2001b]
- $(\sqcap,\sqcup,\neg,\exists R.C,\forall R.C,\leq\geq R)$ $(\sqcap)$ (known as $\mathcal{ALCNR}$) satisfiability w.r.t. a set of axioms (only membership was proved) in [Buchheit *et al.*, 1993a])

3.9.7 Undecidability results

- $\mathcal{FL}^-(\circ, =)$, which is a subset of the language of the knowledge representation system Kl-One [Schmidt-Schauß, 1989] (see Section 3.6.1 for undecidability of $\mathcal{FL}^-(\circ, \subseteq)$)
- $\mathcal{FL}^-(\circ, \subseteq, ^-, \mathit{functionality}, R|_C)$, which is a subset of the language of the knowledge representation system Nikl [Patel-Schneider, 1989a]
- $(), (\sqcap, \circ, \neg)$ (known as U) [Schild, 1989]
- $\mathcal{ALCN}(\circ, \sqcup, ^-)$, $\mathcal{ALCN}(\circ, \sqcap)$ satisfiability w.r.t. a set of axioms [Baader and Sattler, 1999]

Acknowledgements

I thank Franz Baader for useful and stimulating discussions on the proofs of Lemmata 3.34,3.35, and many other comments and help. I am indebted to Maurizio Lenzerini, Daniele Nardi, Werner Nutt and Andrea Schaerf, co-authors of many papers containing results presented in this chapter. I thank also Fabio Massacci for involving me in the experimental evaluation of reasoning algorithms. Giuseppe De Giacomo wasted some time discussing automata with me, and Diego Calvanese promised to make helpful comments on an early draft; I thank them both.

The work has been supported by italian CNR (projets LAICO, DeMAnD, "Metodi di Ragionamento Automatico nella modellazione ed analisi di dominio"), and italian MURST (project MOSES).