Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research Report
RR-91-02

The Complexity of Existential Quantification in Concept Languages

Francesco Donini, Bernhard Hollunder,
Maurizio Lenzerini, Alberto Marchetti Spaccamela,
Daniele Nardi, Werner Nutt

January 1991

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
D-6750 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11, FRG
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, Philips, SEMA Group Systems, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Intelligent Communication Networks
- Intelligent Cooperative Systems.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Prof. Dr. Gerhard Barth
Director

The Complexity of Existential Quantification in Concept Languages

Francesco Donini, Bernhard Hollunder, Maurizio Lenzerini, Alberto Marchetti Spaccamela, Daniele Nardi, Werner Nutt

DFKI-RR-91-02

The Complexity of Existential Quantification in Concept Languages

Francesco M. Donini* Bernhard Hollunder†
Maurizio Lenzerini* Alberto Marchetti Spaccamela‡
Daniele Nardi* Werner Nutt†

Abstract

Much of the research on concept languages, also called terminological languages, has focused on the computational complexity of subsumption. The intractability results can be divided into two groups. First, it has been shown that extending the basic language $\mathcal{FL}^-$ with constructs containing some form of logical disjunction leads to co-NP-hard subsumption problems. Second, adding negation to $\mathcal{FL}^-$ makes subsumption PSPACE-complete.

The main result of this paper is that extending $\mathcal{FL}^-$ with unrestricted existential quantification makes subsumption NP-complete. This is the first proof of intractability for a concept language containing no construct expressing disjunction—whether explicitly or implicitly. Unrestricted existential quantification is therefore, alongside disjunction, a source of computational complexity in concept languages.

*Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Salaria 113, I-00198 Roma, Italy

†German Research Center for Artificial Intelligence (DFKI), Postfach 2080, D-6750 Kaiserslautern, Germany

‡Dipartimento di Matematica, Università de L'Aquila, I-67100 L'Aquila, Italy

Contents

1 Introduction

The interest in concept languages, also called terminological languages, originated from the study of knowledge representation languages, such as KL-ONE [BS85]. In contrast to earlier formalisms like frames and semantic networks, concept languages (see [BL84, Neb90, NS90]) have the advantage of a Tarski style declarative semantics that allows them to be conceived as sublanguages of predicate logic. In these languages, a *concept* is built up of two kinds of symbols, primitive concepts and roles, which can be combined by various language constructs yielding complex concepts. Given a domain, concepts are interpreted as subsets of this domain, and roles are interpreted as binary relations. Different languages are distinguished by the constructs they provide.

Reasoning about concepts is based on *subsumption*: a concept C is subsumed by a concept D if in every interpretation C denotes a subset of the set denoted by D. The subsumption relation implicitly defines a taxonomy of concepts, which is used in designing and running knowledge-based systems. Moreover, subsumption plays a key role in terminological reasoning in that other deductive tasks can be reduced to it. For instance, a subsumption checker can detect *unsatisfiable* concepts (also called incoherent concepts), i.e. concepts which denote the empty set in every interpretation, since a concept is unsatisfiable if and only if it is subsumed by the empty concept $\bot$. Analogously, equivalence and disjointness of two concepts can be reduced to subsumption.

The central role of subsumption in terminological reasoning has motivated the study of its computational complexity in several concept languages. The goal was to identify languages for which the subsumption problem can be computed efficiently while retaining a great expressive power.

Complexity analysis of subsumption originated with the seminal paper by Brachman and Levesque [BL84]. They gave a polynomial algorithm for the small language $\mathcal{FL}^-$, which includes concept conjunction, universal quantification on roles, and a restricted form of existential quantification. They also showed that by adding role restrictions to $\mathcal{FL}^-$, yielding the language called $\mathcal{FL}$, subsumption becomes co-NP-hard. Later, Nebel [Neb88] considered the extension of $\mathcal{FL}^-$ with role conjunctions and number restrictions, which again gives rise to a co-NP-hard subsumption problem.

In a recent paper Schmidt-Schauß and Smolka [SS91] investigate the language $\mathcal{ALC}$, that arises from $\mathcal{FL}^-$ by adding negation of concepts. They provide a decision procedure for $\mathcal{ALC}$ that is based on a calculus of constraints. By exploiting the features of this calculus they proved that subsumption and unsatisfiability in $\mathcal{ALC}$ are PSPACE-complete.

In addition they showed that unsatisfiability is co-NP-complete for a lan-

guage obtained from $\mathcal{FL}^-$ by adding negation of primitive concepts and union of concepts. This result is based on the observation that unsatisfiability in a language with unions and negation of primitive concepts is at least as hard as unsatisfiability in propositional logic. As noticed in [SS91], role restrictions like in $\mathcal{FL}$ contain an implicit form of disjunction. Finally, disjunction arises also from number restrictions [HNS90].

From these results it became clear that the presence of disjunctive constructs, such as union of concepts, role restriction, and number restriction, leads to co-NP-hardness. However, it was unknown whether other constructs would make reasoning intractable. In particular, it was an open problem whether subsumption and unsatisfiability were intractable for the language $\mathcal{ALE}$, which extends $\mathcal{FL}^-$ with unrestricted existential quantification on roles and negation of primitive concepts [SS88].

The key result presented in this paper is that subsumption and unsatisfiability in $\mathcal{ALE}$ are NP-hard. In fact, subsumption is NP-hard for the simpler language $\mathcal{FLE}^-$, which extends $\mathcal{FL}^-$ with unrestricted existential quantification on roles. $\mathcal{FLE}^-$ is the first concept language not including disjunctive constructs to be proved intractable. Moreover, in contrast with the previous intractability results, subsumption in $\mathcal{FLE}^-$ is proved NP-hard rather than co-NP-hard. Therefore, unrestricted existential quantification must be considered, alongside disjunction, a source of computational complexity in concept languages. Notice that since $\mathcal{FLE}^-$ is a sublanguage of $\mathcal{FL}$, subsumption in $\mathcal{FL}$ is not only co-NP-hard, but also NP-hard.

A second result presented in this paper is that subsumption and unsatisfiability in $\mathcal{ALE}$ have the same computational complexity, namely, they can be solved in nondeterministic polynomial time. The NP-easiness of unsatisfiability of $\mathcal{ALE}$ was proved in [SS91]. However, the result on subsumption is somewhat unexpected for the following reason. Since C is subsumed by D if and only if $C \sqcap \neg D$ is unsatisfiable (where $\sqcap$ denotes conjunction of concepts and $\neg$ denotes negation) subsumption can be rephrased in terms of unsatisfiability. If D contains conjunction, which is a basic construct in $\mathcal{ALE}$, then disjunction is introduced into $C \sqcap \neg D$ via negation. Surprisingly, the disjunction present in unsatisfiability problems of this kind does not cause an increase in complexity.

To prove our results we adopt a rule-based calculus for deciding the unsatisfiability of concepts that is similar in spirit to the one used in [SS91] but employs a more concise notation, that points out its similarity to the tableaux calculus for first order predicate logic [Smu68]. In fact, if one translates concepts into logical formulae and applies to them the tableaux calculus with a suitable control strategy, one essentially obtains the calculus described here.

The results reported in the present paper not only solve an open problem

but also provide an intuitive understanding of the computational complexity of concept languages. Informally, we now can identify disjunction and existential quantification as two different sources of complexity. The presence of the former is reflected by co-NP-hardness results, whereas the presence of the latter is expressed in NP-hardness results. The PSPACE-hardness of subsumption in $\mathcal{ALC}$ can then be attributed to the interaction of both.

The above considerations do not take into account another source of complexity in terminological reasoning which already shows up with the most simple languages. Recently, Nebel has proved that subsumption in $\mathcal{FL}^-$ becomes co-NP-hard if concepts are given by a so-called terminology, i.e. a set of definitions of concepts [Neb90]. In the present paper, we assume that concepts are given as expressions, and therefore we do not deal with the problem of handling concept definitions.

The rest of the paper is organized as follows: in the next section we formally define the syntax and the semantics of $\mathcal{FLE}^-$ and $\mathcal{ALE}$, and sketch a calculus for deciding unsatisfiability of $\mathcal{ALE}$-concepts. In Section 3, we show that unsatisfiability in $\mathcal{ALE}$ is an NP-complete problem. In Section 4, we first prove that subsumption in $\mathcal{FLE}^-$ is NP-hard, and then we present a nondeterministic polynomial-time algorithm for subsumption in $\mathcal{ALE}$.

2 Preliminaries

In this section we provide the essential notions about the concept languages considered in this paper; for a general presentation see [NS90].

We start by considering the language $\mathcal{FL}^-$ [BL84], where *concepts* (denoted by the letters C and D) are built out of *primitive concepts* (denoted by the letter A) and *roles* (denoted by the letter R) according to the syntax rules

$$C, D \longrightarrow A \mid \top \mid C \sqcap D \mid \forall R.C \mid \exists R.\top,$$

where $\top$ denotes the universal concept.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$ (the *domain* of $\mathcal{I}$) and a function $\cdot^{\mathcal{I}}$ (the *interpretation function* of $\mathcal{I}$) that maps every concept to a subset of $\Delta^{\mathcal{I}}$ and every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that the following equations are satisfied:

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \forall b.\, (a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.\top)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b.\, (a,b) \in R^{\mathcal{I}}\}.
\end{aligned}$$

An interpretation $\mathcal{I}$ is a *model* for a concept C if $C^{\mathcal{I}}$ is nonempty. A concept is *satisfiable* if it has a model and *unsatisfiable* otherwise. We say that C is *subsumed* by D if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every interpretation $\mathcal{I}$. We say that C is *equivalent* to D if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every interpretation $\mathcal{I}$.

The first extension of $\mathcal{FL}^-$ that we consider results from the addition of a suitable construct for expressing unrestricted existential quantification over roles. The resulting language is called $\mathcal{FLE}^-$. Its syntax is obtained by extending the rules for $\mathcal{FL}^-$ with

$$C, D \longrightarrow \exists R.C.$$

The semantics of the new construct is defined by

$$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.\, (a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}.$$

A further extension to $\mathcal{FLE}^-$ is obtained by providing one symbol for a special primitive concept, namely the empty concept, and by allowing negation of primitive concepts. The resulting language is called $\mathcal{ALE}$. Its complete syntax is as follows:

$$C, D \longrightarrow A \mid \top \mid \bot \mid \neg A \mid C \sqcap D \mid \forall R.C \mid \exists R.C,$$

where the semantics of the additional constructs is defined by

$$\begin{aligned} \bot^{\mathcal{I}} &= \emptyset \\ (\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}. \end{aligned}$$

Notice that an $\mathcal{ALE}$-concept may be unsatisfiable (for example, $A \sqcap \neg A$ is clearly unsatisfiable). On the contrary, every $\mathcal{FLE}^-$-concept is satisfiable, since no negative information can be expressed in this language [SS91].

In $\mathcal{ALE}$ only a restricted form of negation is available. If a language allows for complements of arbitrary concepts, unsatisfiability and subsumption can be reduced to each other. We denote the complement of an arbitrary concept C as $\neg C$, and we interpret $\neg C$ as $\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$. Now, C is unsatisfiable if and only if C is subsumed by $\bot$, and C is subsumed by D if and only if $C \sqcap \neg D$ is unsatisfiable. Notice that, in general, even if C and D are $\mathcal{FLE}^-$-concepts (respectively, $\mathcal{ALE}$-concepts), $C \sqcap \neg D$ is not an $\mathcal{FLE}^-$-concept (resp. $\mathcal{ALE}$-concept).

We now turn to our rule-based calculus for deciding the satisfiability of $\mathcal{ALE}$-concepts. The calculus operates on constraints consisting of variables, concepts, and roles. We assume that there exists an alphabet of variable symbols, which will be denoted by the letters x, y, and z. A constraint is a syntactic object of one of the forms

$$x{:}C, \quad xRy,$$

where C is a concept and R is a role.

Let $\mathcal{I}$ be an interpretation. An *$\mathcal{I}$-assignment* α is a function that maps every variable to an element of $\Delta^{\mathcal{I}}$. We say that α *satisfies* $x{:}\,C$ if $\alpha(x) \in C^{\mathcal{I}}$, and α *satisfies* xRy if $(\alpha(x), \alpha(y)) \in R^{\mathcal{I}}$. A constraint c is *satisfiable* if there is an interpretation $\mathcal{I}$ and an $\mathcal{I}$-assignment α such that α satisfies c. A *constraint system* S is a finite, nonempty set of constraints. An $\mathcal{I}$-assignment α *satisfies* a constraint system S if α satisfies every constraint in S. A constraint system S is *satisfiable* if there is an interpretation $\mathcal{I}$ and an $\mathcal{I}$-assignment α such that α satisfies S.

The following proposition as well as the other propositions in this section can easily be derived from the results reported in [SS91].

Proposition 2.1 *A concept C is satisfiable if and only if the constraint system $\{x{:}\,C\}$ is satisfiable.*

Our calculus starts with a constraint system $S = \{x{:}\,C\}$, and, in subsequent steps, constraints are added to S according to a set of propagation rules, until either a contradiction is generated or a model of C can be obtained from the resulting system. The *propagation rules* are:

1. $S \rightarrow_{\sqcap} \{x{:}\,C_1,\ x{:}\,C_2\} \cup S$
   if $x{:}\,C_1 \sqcap C_2$ is in S, and either $x{:}\,C_1$ or $x{:}\,C_2$ is not in S
2. $S \rightarrow_{\exists} \{xRy,\ y{:}\,C\} \cup S$
   if $x{:}\,\exists R.C$ is in S, there is no z such that both xRz and $z{:}\,C$ are in S, and y is a new variable
3. $S \rightarrow_{\forall} \{y{:}\,C\} \cup S$
   if $x{:}\,\forall R.C$ is in S, xRy is in S, and $y{:}\,C$ is not in S.

Proposition 2.2 *Let S be a constraint system. If S' is obtained from S by the application of a propagation rule, then S is satisfiable if and only if S' is satisfiable.*

A constraint system is said to be *complete* if no propagation rule applies to it. Note that, up to variable renaming, only one complete constraint system can be derived from $\{x{:}\,C\}$. A *clash* is a constraint system having either the form $\{x{:}\,\bot\}$ or the form $\{x{:}\,A,\ x{:}\,\neg A\}$.

Proposition 2.3 *If C is an $\mathcal{ALE}$-concept, then after finitely many applications of the propagation rules one obtains a complete constraint system S from $\{x{:}\,C\}$. Moreover, S is satisfiable if and only if it contains no clash.*

By virtue of the above properties, the calculus can be easily turned into a decision procedure that checks an $\mathcal{ALE}$-concept C for unsatisfiability by verifying whether the complete constraint system derived from $\{x\!:\!C\}$ contains a clash.

Notice that, due to the form of the $\rightarrow_\exists$-rule, the number of variables generated by the procedure may be exponential in the size of C. For example, it is easy to see that the complete system derived from $\{x\!:\!C\}$, where C is of the form

$$\exists R.C_1 \sqcap \exists R.C_1' \sqcap \forall R.(\exists R.C_2 \sqcap \exists R.C_2' \sqcap \forall R.(\ldots(\exists R.C_n \sqcap \exists R.C_n' \sqcap \forall R.D)\ldots)),$$

contains at least $2^n + 1$ variables.

3 Unsatisfiability in $\mathcal{ALE}$

In this section we prove that unsatisfiability in $\mathcal{ALE}$ is an NP-complete problem. In order to show this, we need to refine the notion of constraint system by identifying particular subsets of a constraint system called traces. Traces are built using trace rules, which are defined as follows.

The *trace rules* consist of the $\rightarrow_\sqcap$-rule and the $\rightarrow_\forall$-rule given in the previous section, together with the rule

$$S \rightarrow_{T\exists} \{xRz,\ y\!:\!C\} \cup S$$

if $x\!:\!\exists R.C$ is in S, there is no constraint of the form xRz in S, and y is a new variable.

The difference between the $\rightarrow_\exists$-rule and the $\rightarrow_{T\exists}$-rule is that the latter is applied only once for a variable x. We are thus compelled to make a nondeterministic choice amongst the constraints of the form $x\!:\!\exists R.C$.

Let C be an $\mathcal{ALE}$-concept. A constraint system T is a *trace* of $\{x\!:\!C\}$ if T is obtained from $\{x\!:\!C\}$ by the application of the trace rules. It follows from the results in [SS91] that C is unsatisfiable if and only if there exists a trace of $\{x\!:\!C\}$ that contains a clash. Therefore, to prove that a concept is unsatisfiable, it is sufficient to guess one trace containing a clash out of the set of all possible traces of $\{x\!:\!C\}$. Since the size of a trace of $\{x\!:\!C\}$ is bounded polynomially by the size of C, it follows that there exists a nondeterministic polynomial-time algorithm for checking the unsatisfiability of $\mathcal{ALE}$-concepts.

To prove that deciding the unsatisfiability of $\mathcal{ALE}$-concepts is an NP-hard problem we introduce the following *set traversal problem.* A *positive clause* is a finite set of positive integers. If $\mathcal{M} = \{M_1, \ldots, M_m\}$ is a finite set of positive clauses, then a *traversal* of $\mathcal{M}$ is a finite set of positive integers N such that $N \cap M_l$ is a singleton for $l \in 1..m$. The set traversal problem is

defined as follows: given a finite set $\mathcal{M}$ of positive clauses, decide whether $\mathcal{M}$ has a traversal or not.

Theorem 3.1 *The set traversal problem is NP-complete.*

Proof. It is easy to see that the problem is a reformulation of ONE-IN-THREE 3SAT (see [GJ79], p. 259). □

We now show that the set traversal problem is reducible to unsatisfiability of $\mathcal{ALE}$-concepts in polynomial time. The reduction consists in associating to every set of positive clauses $\mathcal{M}$ an $\mathcal{ALE}$-concept $C_{\mathcal{M}}$ such that $\mathcal{M}$ has a traversal if and only if $C_{\mathcal{M}}$ is unsatisfiable. The key idea is to relate the traversals of $\mathcal{M}$ to those traces of $\{x\!:\!C_{\mathcal{M}}\}$ that contain a clash.

In the following we assume R to be a fixed role. Let $\mathcal{M} = \{M_1, \ldots, M_m\}$ be a set of positive clauses, and let n be the maximum of the numbers occurring in the clauses of $\mathcal{M}$. We translate $\mathcal{M}$ into the concept

$$C_{\mathcal{M}} = C_1^1 \sqcap \ldots \sqcap C_1^n \sqcap D_1.$$

The C_1^j's are inductively defined through the equations

$$C_l^j = \begin{cases} \exists R.C_{l+1}^j & \text{if } j \in M_l \\ \forall R.C_{l+1}^j & \text{if } j \notin M_l \end{cases} \quad \text{and} \quad C_{m+l}^j = \begin{cases} \exists R.C_{m+l+1}^j & \text{if } j \in M_l \\ \forall R.C_{m+l+1}^j & \text{if } j \notin M_l \end{cases}$$

for $l \in 1..m$ and by the equation $C_{2m+1}^j = \top$. The concept D_1 is defined by the equations

$$D_l = \forall R.D_{l+1} \qquad \text{and} \qquad D_{m+l} = \forall R.D_{m+l+1}$$

for $l \in 1..m$ and by the equation $D_{2m+1} = \bot$.

Observe that for every number j occurring in $\mathcal{M}$ there is a corresponding concept C_1^j, and for every clause $M_l \in \mathcal{M}$ there are corresponding levels l and $m+l$ in the C_1^j's. The number j is present in the lth clause if and only if there is an existential quantifier in the concept C_1^j at level l and at level $m+l$. As we shall see later the two layered construction of $C_{\mathcal{M}}$ is crucial for the correctness of the reduction. The concept D_1 is designed in such a way that a clash for $\{x\!:\!C_{\mathcal{M}}\}$ can only occur in a trace at level $2m+1$.

As an example, consider the following instance of the set traversal problem:

$$\mathcal{M} = \{\{1,3,5\},\{2,4\},\{4,5\}\}.$$

The corresponding $\mathcal{ALE}$-concept $C_{\mathcal{M}}$ is given by the conjunction of

$$C_1^1 \;=\; \exists R.\forall R.\forall R.\exists R.\forall R.\forall R.\top$$

$$
\begin{aligned}
C_1^2 &= \forall R.\exists R.\forall R.\forall R.\exists R.\forall R.\top \\
C_1^3 &= \exists R.\forall R.\forall R.\exists R.\forall R.\forall R.\top \\
C_1^4 &= \forall R.\exists R.\exists R.\forall R.\exists R.\exists R.\top \\
C_1^5 &= \exists R.\forall R.\exists R.\exists R.\forall R.\exists R.\top \\
D_1 &= \forall R.\forall R.\forall R.\forall R.\forall R.\forall R.\bot.
\end{aligned}
$$

Notice that 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 one object to belong to the extension of $\bot$. As we said before, the idea of the reduction is to create a correspondence between such a trace and a traversal of $\mathcal{M}$.

In order to formally characterize such a correspondence, we need to define the notion of 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 variables y, z such that T contains the constraints $y\colon C$, yRz, and $z\colon D$. Therefore, an existentially quantified concept $\exists R.D$ is active in T if the $\rightarrow_\exists$-rule has been applied to some constraint $y\colon \exists R.D$ in T. Intuitively, if C_k^j is active in a trace of $\{x\colon C_\mathcal{M}\}$ containing a clash, then j belongs to a traversal of $\mathcal{M}$.

Lemma 3.2 *Let T be a trace of $\{x\colon C_\mathcal{M}\}$.*

1. *Suppose C_k^j is active in T. Then for all $l \in 1..k$ the concept C_l^j is active in T if it is of the form $\exists R.C_{l+1}^j$.*

2. *If T contains a clash, then for every $l \in 1..2m$ there exists exactly one j such that C_l^j is active in T.*

Proof. 1. It suffices to show the following for every $k \in 1..(2m+1)$: if T contains the constraint $y_k\colon C_k^j$ for some variable y_k then T contains constraints $y_l\colon C_l^j$ and $y_l R y_{l+1}$ for all $l \in 1..(k-1)$.

We proceed by induction on k. For $k = 1$ the claim trivially holds. Suppose the claim holds for a given k. Assume that $y_{k+1}\colon C_{k+1}^j$ is in T. We distinguish two cases.

If $C_k^j = \exists R.C_{k+1}^j$, then the constraint $y_{k+1}\colon C_{k+1}^j$ has been introduced by the $\rightarrow_{T\exists}$-rule. From the conditions of application of this rule, it follows that the constraint $y_k\colon C_k^j$ is in T for some variable y_k. From the inductive hypothesis we conclude that there are constraints $y_l\colon C_l^j$ and $y_l R y_{l+1}$ in T for all $l \in 1..(k-1)$. In addition, the application of the $\rightarrow_{T\exists}$-rule has also introduced the constraint $y_k R y_{k+1}$.

If $C_k^j = \forall R.C_{k+1}^j$, then the constraint $y_{k+1}\colon C_{k+1}^j$ has been introduced by the $\rightarrow_\forall$-rule. From the conditions of application of this rule, it follows that

the constraints $y_k: C_k^j$ and $y_k R y_{k+1}$ are in T for some variable y_k. Together with the inductive hypothesis this yields the claim for $k+1$.

2. From the definition of traces it is easy to see that there is at most one j such that C_l^j is active in T. It remains to show that there is at least one such j. Assume on the contrary that for some $l \in 1..2m$ there is no j such that C_l^j is active in T. Then for every $k \in (l+1)..(2m+1)$ there is no constraint in T of the form $y: C_k^i$ or $y: D_k$. Hence T does not contain the only possible clash $y: D_{2m+1}$. □

Using the above properties, we are able to prove the correctness of our reduction.

Theorem 3.3 *A set $\mathcal{M}$ of positive clauses has a traversal if and only if $C_{\mathcal{M}}$ is unsatisfiable.*

Proof. Suppose $\mathcal{M} = \{M_1, \ldots, M_m\}$ is a set of positive clauses and $C_{\mathcal{M}} = C_1^1 \sqcap \ldots \sqcap C_1^n \sqcap D_1$ is its translation.

"$\Rightarrow$" Let N be a traversal of $\mathcal{M}$. To prove that $C_{\mathcal{M}}$ is unsatisfiable, we show that there exists a trace T of $\{x_1: C_{\mathcal{M}}\}$ that contains a clash. To obtain T we inductively define a sequence of traces T_l of $\{x_1: C_{\mathcal{M}}\}$, where $l \in 1..(2m+1)$, such that $T_l \subseteq T_{l+1}$ and $T_{2m+1} = T$. Let

$$T_1 = \{x_1: C_1^j \mid j \in N\} \cup \{x_1: D_1\}$$

and

$$T_{l+1} = T_l \cup \{x_l R x_{l+1}\} \cup \{x_{l+1}: C_{l+1}^j \mid j \in N\} \cup \{x_{l+1}: D_{l+1}\}.$$

Obviously, $T = T_{2m+1}$ contains a clash, because $D_{2m+1} = \bot$.

To show that T is a trace we prove that T_{l+1} can be obtained from T_l by application of the trace rules. Since N is a traversal, there is exactly one $k \in N$ such that $C_l^k = \exists R.C_{l+1}^k$. Hence,

$$T_l \rightarrow_{T\exists} T_l' = T_l \cup \{x_l R x_{l+1},\ x_{l+1}: C_{l+1}^k\}$$

by application of the $\rightarrow_{T\exists}$-rule. Since we have $C_l^j = \forall R.C_{l+1}^j$ for $j \in N \setminus \{k\}$ and $D_l = \forall R.D_{l+1}$, it is easy to see that T_{l+1} can be obtained from T_l' by application of the $\rightarrow_\forall$-rule.

"$\Leftarrow$" If $C_{\mathcal{M}}$ is unsatisfiable, then there exists a trace T of $\{x: C_{\mathcal{M}}\}$ such that T contains a clash and for every $l \in 1..2m$ there exists exactly one j such that C_l^j is active in T.

Let

$$N = \{j \mid C_{m+l}^j \text{ is active in } T \text{ for some } l \in 1..m\}.$$

We show that N is a traversal. Let M_l be a clause in $\mathcal{M}$.

For every $l \in 1..m$, there exists a j such that C^j_{m+l} is active in T. Hence we have $j \in N$. Furthermore, $C^j_{m+l} = \exists R.C^j_{m+l+1}$ by the definition of active concepts. By construction of $C_{\mathcal{M}}$, we have that $j \in M_l$. Thus $j \in N \cap M_l$.

Suppose there are i, j such that $i, j \in N \cap M_l$. Now we exploit the two-layered construction of $C_{\mathcal{M}}$. Since $i, j \in N$, by definition of N there are h, k such that C^i_{m+h} and C^j_{m+k} are active in T. Since $i, j \in M_l$, by construction of $C_{\mathcal{M}}$ we have $C^i_l = \exists R.C^i_{l+1}$ and $C^j_l = \exists R.C^j_{l+1}$. By 3.2.1, we know that C^i_l and C^j_l are active in T. Hence $i = j$ by 3.2.2. □

We can now state the main theorem of this section.

Theorem 3.4 *Unsatisfiability in $\mathcal{ALE}$ is NP-complete.*

Proof. The claim follows from the existence of a nondeterministic polynomial algorithm for the problem (see [SS91]), and from the observation that, for every set of positive clauses $\mathcal{M}$, the concept $C_{\mathcal{M}}$ is in $\mathcal{ALE}$. □

4 Subsumption in $\mathcal{FLE}^-$ and $\mathcal{ALE}$

In this section we show that subsumption is NP-complete both in $\mathcal{ALE}$ and $\mathcal{FLE}^-$. Let us first consider NP-hardness.

Theorem 4.1 *Subsumption is NP-hard both in $\mathcal{ALE}$ and $\mathcal{FLE}^-$.*

Proof. The claim holds with regard to $\mathcal{ALE}$, since $\mathcal{ALE}$ contains the empty concept $\bot$ and therefore unsatisfiability is a special case of subsumption.

For $\mathcal{FLE}^-$, observe that $C_{\mathcal{M}} = C^1_1 \sqcap \ldots \sqcap C^n_1 \sqcap D_1$ is unsatisfiable if and only if $C^1_1 \sqcap \ldots \sqcap C^n_1$ is subsumed by $\neg D_1$. The claim holds, since $C^1_1 \sqcap \ldots \sqcap C^n_1$ is in $\mathcal{FLE}^-$ and $\neg D_1$ can be rewritten to the equivalent concept E_1, inductively defined by

$$E_l = \exists R.E_{l+1} \qquad \text{and} \qquad E_{m+l} = \exists R.E_{m+l+1}$$

for $l \in 1..m$ and $E_{2m+1} = \top$. Obviously, E_1 is in $\mathcal{FLE}^-$. □

Let us now turn to the NP-easiness of subsumption in $\mathcal{ALE}$ and $\mathcal{FLE}^-$. Since $\mathcal{FLE}^-$ is a sublanguage of $\mathcal{ALE}$, it is sufficient to consider subsumption in $\mathcal{ALE}$. In order to prove that subsumption in $\mathcal{ALE}$ can be solved in nondeterministic polynomial time, we reduce subsumption between D and C to unsatisfiability of $C \sqcap \neg D$.

In general, the concept $C \sqcap \neg D$ is not in $\mathcal{ALE}$, since $\mathcal{ALE}$ does not allow for negation of arbitrary concepts. To cope with arbitrary negations

we consider the language $\mathcal{ALC}$ [SS91], which is an extension of $\mathcal{ALE}$ with complements and unions:

$$C, D \longrightarrow \neg C \mid C \sqcup D.$$

As already mentioned, the interpretation of complements is given by $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$. Unions are interpreted such that $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$.

As in [SS91] we single out a special class of concepts as normal forms: a concept C is called *simple* if negation inside C occurs only in front of primitive concepts, i.e. negations have the form $\neg A$. An arbitrary $\mathcal{ALC}$-concept can be transformed in linear time into an equivalent simple concept by means of the following rewriting rules:

$$\begin{aligned}
\neg\top &\longrightarrow \bot \\
\neg\bot &\longrightarrow \top \\
\neg\neg C &\longrightarrow C \\
\neg(C \sqcap D) &\longrightarrow \neg C \sqcup \neg D \\
\neg(C \sqcup D) &\longrightarrow \neg C \sqcap \neg D \\
\neg(\forall R.C) &\longrightarrow \exists R.\neg C \\
\neg(\exists R.C) &\longrightarrow \forall R.\neg C.
\end{aligned}$$

To check satisfiability of simple $\mathcal{ALC}$-concepts a new rule dealing with union must be added to the propagation rules 1–3.

4. $S \rightarrow_{\sqcup} \{x\!:\!D\} \cup S$

   if $x\!:\!C \sqcup C'$ is in S, neither $x\!:\!C$ nor $x\!:\!C'$ is in S, and $D = C$ or $D = C'$.

In contrast to the rules 1–3, the $\rightarrow_{\sqcup}$-rule is nondeterministic. Therefore several complete constraint systems can be derived from $\{x\!:\!C\}$ if C is a simple $\mathcal{ALC}$-concept that contains unions. To decide the satisfiability of C all complete constraint systems that can be derived from $\{x\!:\!C\}$ must be inspected, which are—up to variable renaming—finitely many. From now on we implicitly refer to the set of rules 1–4 as *propagation rules.* The next proposition follows immediately from results in [SS91].

Proposition 4.2 *Let C be a simple $\mathcal{ALC}$-concept. Then C is satisfiable if and only if there is a complete clash free constraint system that can be derived from S with the propagation rules.*

Based on the above theorem, in [SS91] a PSPACE-algorithm is presented for the satisfiability problem in $\mathcal{ALC}$. The basic idea underlying the algorithm is to generate traces, which require polynomial space, rather than

complete constraint systems, that may have exponential size. The $\mathcal{ALC}$-trace rules consist of the trace rules defined in the previous section together with the $\rightarrow_\sqcup$-rule. From now on we implicitly refer to $\mathcal{ALC}$-trace rules as *trace rules* and to constraint systems generated by means of these rules as *traces*.

As shown in [SS91], satisfiability in $\mathcal{ALC}$ is PSPACE-complete and hence we do not expect it to be solvable in nondeterministic polynomial time. Intuitively, PSPACE-hardness can be explained by the presence of two sources of complexity: unrestricted existential quantification, that may lead to complete constraint systems of exponential size, and union, that may require the generation of an exponential number of complete constraint systems.

In the sequel we show that, when C and D are $\mathcal{ALE}$-concepts, due to the special form of the $\mathcal{ALC}$-concept $C \sqcap \neg D$, union does not act as an additional source of complexity. For this purpose we set up the *s*-rule calculus that operates on *sets* of traces and behaves nondeterministically only for existential quantification. It is interesting to observe that the *s*-rules provide an alternative method for deciding the satisfiability of $\mathcal{ALC}$-concepts.

The *s*-rules are ($\mathcal{T}$ denotes a set of traces):

1. $\{T\} \cup \mathcal{T} \rightarrow^s_* \{T'\} \cup \mathcal{T}$

   if $T \notin \mathcal{T}$ and $T \rightarrow_* T'$ where $* \in \{\sqcap, T\exists, \forall\}$

2. $\{T\} \cup \mathcal{T} \rightarrow^s_\sqcup \{T', T''\} \cup \mathcal{T}$

   if $T \notin \mathcal{T}$, $x{:}\,C \sqcup C'$ is in T, neither $x{:}\,C$ nor $x{:}\,C'$ is in T and $T' = \{x{:}\,C\} \cup T$, $T'' = \{x{:}\,C'\} \cup T$.

The *s*-rules eliminate the nondeterminism introduced by unions, since the two traces that can be obtained by an application of the $\rightarrow_\sqcup$-rule are put both into the new set of traces. The nondeterminism in choosing a constraint to which the $\rightarrow_{T\exists}$-rule applies persists. The following lemma states two properties of the *s*-rule calculus that will be used in the sequel.

Lemma 4.3 *Let C be a simple $\mathcal{ALC}$-concept.*

1. *Every s-rule derivation starting with $\{\{x{:}\,C\}\}$ terminates.*
2. *Suppose that $\mathcal{T}$ has been derived from $\{\{x{:}\,C\}\}$ with the s-rules. Then for every complete constraint system S derived from $\{x{:}\,C\}$ with the propagation rules there is some $T \in \mathcal{T}$ such that—up to variable renaming—T is a subset of S.*

Proof. 1. Follows from the termination of the propagation calculus.
2. Follows by induction on the length of the derivation. □

Next we prove correctness and completeness of the *s*-calculus, i.e. we show that a simple $\mathcal{ALC}$-concept C is unsatisfiable if and only if $\{\{x{:}\,C\}\}$

can be transformed by the s-rules into a set $\mathcal{T}$ such that each trace in $\mathcal{T}$ contains a clash. The critical steps in an s-rule derivation are those using the $\rightarrow^S_{T\exists}$-rule, since they introduce nondeterminism. The idea behind the completeness proof is to use the clashes in complete constraint systems to guide the application of the s-rules.

Let S be a constraint system and x be a variable occurring in S. A variable y is a *successor of x in S* if S contains constraints of the form xR_1y_1, $y_1R_2y_2, \ldots, y_{n-1}R_ny$, where $n \geq 0$. Note that every variable in S is a successor of itself. We say that x *leads to a clash in S* if x has a successor y in S such that S contains either the constraint $y\colon \bot$ or the constraints $y\colon A$, $y\colon \neg A$.

Let C be a simple unsatisfiable $\mathcal{ALC}$-concept and let $\mathcal{S}$ be the set of all complete constraint systems obtainable from $\{x\colon C\}$ with the propagation rules. Since C is unsatisfiable, every $S \in \mathcal{S}$ contains a clash. Suppose $\mathcal{T}$ is a set of traces that are derived from $\{\{x\colon C\}\}$. We say that $\mathcal{T}$ is *covering* if for every $T \in \mathcal{T}$ and $S \in \mathcal{S}$ with $T \subseteq S$ every variable occurring in T leads to a clash in S.

Lemma 4.4 *Let C be a simple unsatisfiable $\mathcal{ALC}$-concept. Suppose that $\mathcal{T}$ has been obtained from $\{\{x\colon C\}\}$ by means of the s-rules. If $\mathcal{T}$ is covering and some s-rule is applicable to $\mathcal{T}$, then there is some $\mathcal{T}'$ such that $\mathcal{T} \rightarrow^S \mathcal{T}'$ and $\mathcal{T}'$ is covering.*

Proof. Suppose $\mathcal{T}$ is covering and $\mathcal{T}'$ is derived from $\mathcal{T}$ by application of the $\rightarrow^S_{\sqcap}$-, $\rightarrow^S_{\forall}$-, or $\rightarrow^S_{\sqcup}$-rule. Then $\mathcal{T}'$ is covering, since neither of these rules introduces new variables.

Suppose the $\rightarrow^S_{T\exists}$-rule is applicable to $\mathcal{T}$, and the $\rightarrow^S_{\sqcap}$-, $\rightarrow^S_{\forall}$-, and $\rightarrow^S_{\sqcup}$-rules are not applicable. Then there is a $T \in \mathcal{T}$ such that $y\colon \exists R.D$ is in T and no constraint yRz is in T.

Let S be a complete constraint system such that $T \subseteq S$. The $\rightarrow_{\sqcap}$-, $\rightarrow_{\forall}$-, and $\rightarrow_{\sqcup}$-rules are not applicable to T. Therefore every constraint of the form $y\colon E$ contained in S is also contained in T.

Let $y\colon \exists R_1.D_1, \ldots, y\colon \exists R_n.D_n$ be all constraints of the form $y\colon \exists R.D$ occurring in T. For $i \in 1..n$ let $T_i = T \cup \{yR_iz_i,\ z_i\colon D_i\}$, where $z_1, \ldots, z_n$ are distinct new variables. We claim that there is a T_i such that for every complete constraint system S with $T_i \subseteq S$ the variable z_i leads to a clash in S.

Assume that this is not the case. Then there exist complete constraint systems $S_1, \ldots, S_n$ with $T_i \subseteq S_i$ such that z_i does not lead to a clash in S_i. For $i \in 1..n$ let $S'_i \subseteq S_i$ be the constraint system that consists of all constraints in S_i in which a successor of z_i occurs. Obviously, S'_i is complete and clash free. Next, consider an arbitrary complete constraint system S with $T \subseteq S$. Let S' be obtained from S by removing all constraints in which a successor

of y occurs that is different from y. Define $S'' = S' \cup S'_1 \cup \ldots \cup S'_n$. The system S'' is complete and contains T. By construction, y is a variable in T that does not lead to a clash in S''. We thus have a contradiction to our assumption, since $\mathcal{T}$ is not covering.

Hence there is some T_i with the desired property. Let $\mathcal{T}'$ be obtained from $\mathcal{T}$ by replacing T with T_i. Then $\mathcal{T}'$ is covering and $\mathcal{T} \rightarrow^{s}_{T\exists} \mathcal{T}'$. □

Using the above lemma we can now prove correctness and completeness of the s-rules.

Theorem 4.5 *Let C be a simple $\mathcal{ALC}$-concept. Then C is unsatisfiable if and only if $\{\{x{:}\,C\}\}$ can be transformed by the s-rules into a set $\mathcal{T}$ such that each trace in $\mathcal{T}$ contains a clash.*

Proof. "⇒" Suppose $\mathcal{T}$ has been derived from $\{\{x{:}\,C\}\}$ by means of the s-rules. If every element of $\mathcal{T}$ contains a clash, then by Lemma 4.3 every complete constraint system derived from $\{x{:}\,C\}$ contains a clash. By Theorem 4.2, the concept C is unsatisfiable.

"⇐" Suppose C is unsatisfiable. Then every complete constraint system derived from $\{x{:}\,C\}$ contains a clash. Hence $\{\{x{:}\,C\}\}$ is covering by definition. Since every s-rule derivation terminates, Lemma 4.4 guarantees that we can derive from $\{\{x{:}\,C\}\}$ a set $\mathcal{T}$ of traces that is covering and to which no s-rule applies.

Let T be a trace in such a $\mathcal{T}$ and S be a complete constraint system derived from $\{x{:}\,C\}$, with $T \subseteq S$. Since no s-rule is applicable to $\mathcal{T}$ there exists in T a variable y that has no other successor in S than itself. For the same reason, every constraint of the form $y{:}\,D$ contained in S is also contained in T. Since y leads to a clash in S, the system S contains the constraint $y{:}\,\bot$ or constraints of the form $y{:}\,A$, $y{:}\,\neg A$. These constraints are also contained in T. Since T was an arbitrary element of $\mathcal{T}$, every element of $\mathcal{T}$ contains a clash. □

The above theorem directly suggests a nondeterministic method for deciding the unsatisfiability of an $\mathcal{ALC}$-concept C. The method generates a number of traces that is—in the general case—exponential in the size of C. This is not surprising, since unsatisfiability in $\mathcal{ALC}$ is PSPACE-complete.

When checking subsumption between $\mathcal{ALE}$-concepts D and C, though, a better result can be achieved: in particular, we now show that the number of traces generated by the application of s-rules is bounded by the size of D. Since traces are of polynomial size, the s-rules provide a nondeterministic polynomial time method for checking subsumption in $\mathcal{ALE}$.

The proof heavily relies on the structure of traces that arise when checking subsumption. A constraint in a trace T is *closed* if it is of the form

- $x{:}\, C \sqcap D$, and both $x{:}\, C$ and $x{:}\, D$ are in T;
- $x{:}\, \forall R.C$, and for all y such that xRy is in T, $y{:}\, C$ is in T;
- $x{:}\, \exists R.C$, and there exist y, R' such that $xR'y$ is in T;
- $x{:}\, C \sqcup D$, and either $x{:}\, C$ or $x{:}\, D$ is in T.

Intuitively, a constraint is closed if no trace rule applies to it. A constraint in T is *open* if it is not closed.

We say that a concept C *contains intersections* or *unions* if the symbols "$\sqcap$" or "$\sqcup$", respectively, occur in the string C. Note that if D is an $\mathcal{ALE}$-concept, then by rewriting $\neg D$ into a simple concept we obtain a concept D' that contains no intersection. Moreover, $\{x{:}\, C \sqcap D'\}$ is unsatisfiable if and only if $\{x{:}\, C, x{:}\, D'\}$ is unsatisfiable.

Lemma 4.6 *Let C, D be $\mathcal{ALE}$-concepts and let D' be obtained by rewriting $\neg D$ to a simple $\mathcal{ALC}$-concept. Let T be a trace derived from $\{x{:}\, C,\ x{:}\, D'\}$. Then there is at most one open constraint $y{:}\, E$ in T such that E contains unions. If there is such a constraint then E contains no intersections.*

Proof. By induction on the lenght of the derivation. The base case is obvious, since C contains no unions and D' contains no intersections. Let T be a trace satisfying the inductive hypothesis. We show that every trace T' obtained from T by applying a trace rule to some constraint $y{:}\, E$ satisfies the inductive hypothesis, too.

Suppose the $\rightarrow_{\sqcap}$-rule is applied to $y{:}\, E \sqcap E'$. Then neither E nor E' contains unions, since $E \sqcap E'$ contains an intersection. Hence neither of the new constraints $y{:}\, E$, $y{:}\, E'$ contains unions.

Suppose the $\rightarrow_{T\exists}$-rule is applied to $y{:}\, \exists R.E$, introducing yRz and $z{:}\, E$. If E contains no unions then we are done. Otherwise, $y{:}\, \exists R.E$ is the only open constraint in T that contains unions. Moreover, $\exists R.E$ contains no intersections. In T' the constraint $y{:}\, \exists R.E$ is closed and $z{:}\, E$ is the only open constraint that contains unions. Moreover, E contains no intersections.

Suppose the $\rightarrow_{\forall}$-rule is applied to $y{:}\, \forall R.E$ and yRz, introducing $z{:}\, E$. Then $y{:}\, \forall R.E$ is open in T. If $y{:}\, \forall R.E$ contains unions, then there is no other open constraint containing unions. Since T is a trace, z is the only variable such that yRz is in T. Hence, $y{:}\, \forall R.E$ is closed in T' and $z{:}\, E$ is the only open constraint in T' containing unions. Moreover, E contains no intersections, since $\forall R.E$ contains no intersections.

Suppose the $\rightarrow_{\sqcup}$-rule is applied to $y{:}\, E \sqcup E'$. Then $y{:}\, E \sqcup E'$ is the only open constraint in T containing unions. Hence, in the resulting system there is at most one open constraint containing unions. Since $E \sqcup E'$ contains no intersections, neither E nor E' contains intersections. □

From the preceding results we can conclude that checking subsumption of $\mathcal{ALE}$-concepts is NP-easy.

Theorem 4.7 *Subsumption in $\mathcal{ALE}$ can be decided in nondeterministic polynomial time.*

Proof. Let C, D be $\mathcal{ALE}$-concepts and let D' be obtained by rewriting $\neg D$ to a simple $\mathcal{ALC}$-concept. By Theorem 4.5, C is subsumed by D if and only if $\{\{x\!:C, x\!:D'\}\}$ can be transformed with the s-rules into a set of traces each of which contains a clash. Since traces are of polynomial size, it suffices to show that any s-rule derivation starting with $\{\{x\!:C, x\!:D'\}\}$ leads to a set of traces $\mathcal{T}$ whose cardinality is bounded by the size of D'.

In fact, the only s-rule increasing the number of traces is the $\rightarrow^{s}_{\sqcup}$-rule. This rule can only be applied to an open constraint containing unions. By Lemma 4.6, its application decreases the number of times that the union symbol occurs in open constraints at least by one. Therefore in any derivation, the $\rightarrow^{s}_{\sqcup}$-rule can be applied at most as many times as the union symbol occurs in D'. □

Corollary 4.8 *Subsumption in $\mathcal{ALE}$ and in $\mathcal{FLE}^-$ is NP-complete.*

5 Conclusion

In this paper we have considered the subsumption and the unsatisfiability problem for two concept languages that extend the basic language $\mathcal{FL}^-$ [BL84].

We have proved that subsumption is NP-complete for the language $\mathcal{FLE}^-$ that extends $\mathcal{FL}^-$ with unrestricted existential quantification. Since existential quantification can be realised through role restriction, Brachman and Levesque's language $\mathcal{FL}$ contains $\mathcal{FLE}^-$ as a sublanguage. We thus have complemented their result that subsumption in $\mathcal{FL}$ is co-NP-hard [BL84] by showing that it is also NP-hard.

In addition, we have shown that both unsatisfiability and subsumption are NP-complete for the language $\mathcal{ALE}$ that is obtained from $\mathcal{FLE}^-$ by adding negation of primitive concepts. The question of the hardness of unsatisfiability in $\mathcal{ALE}$ was raised by Schmidt-Schauß and Smolka [SS88], and its answer provides a missing link among various complexity results for concept languages. Former work identified disjunctive constructs that together with concept conjunction give rise to intractability [BL84, Neb88, SS91], whereas we have shown that the interplay of universal and existential quantifiers is a cause of complexity, too. Therefore we conclude that in concept

languages there are two sources of complexity whose different nature is illustrated by the fact that the first makes subsumption co-NP-hard and the latter makes it NP-hard. The combination of them in the language $\mathcal{ALC}$ introduced in [SS91] intuitively explains the PSPACE-completeness of subsumption in $\mathcal{ALC}$. Based on these results it has been possible to classify from the point of view of the computational complexity of deduction a large set of concept languages obtained by combining the most well-known constructs [DLNN91].

Originally, complexity analysis of terminological reasoning was set up with the goal to identify languages for which subsumption can be decided in polynomial time [BL84]. Now it has turned out that practically all interesting constructs in concept languages lead to intractability, and even the most modest languages are affected by this problem when the use of terminologies is allowed—which is often the case in implemented systems.

One might conclude from these results that terminological reasoning in all its variants is infeasible. Such a conclusion would implicitly assume that the complexity analysis of subsumption is intended to restrict the practical use of concept languages to those where subsumption can be computed in polynomial time. However, it is our opinion that the study of the complexity of concept languages goes far beyond a mere classification of tractable and intractable languages.

First of all, the results developed so far refer to the computational complexity in the worst case, which represents only one aspect to be taken into account when considering the practical use of concept languages. Notice that, as pointed out in [Neb90], another aspect that deserves further investigation is the characterization of the average cases occurring in practice. Second, the techniques used for the complexity analysis have provided the formal basis for the design of effective algorithms for computing subsumption and unsatisfiability in a large class of concept languages [HNS90]. Finally, in the design of deduction procedures for knowledge representation systems based on concept languages, one can take advantage of the knowledge about the complexity of subsumption, by isolating difficult cases and using specialized efficient algorithms whenever possible.

For all the above reasons, we believe that the research on the computational complexity of concept languages is a valuable support for the design of knowledge-based systems embedding forms of terminological reasoning.

Acknowledgements

This work was partly funded by the ESPRIT Basic Research Action 3012 (Compulog), ESPRIT Basic Research Action 3075 (Alcom), the Italian CNR under Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, and the German Bundesministerium für Forschung und Technologie under grant ITW 8903 0.

References

[BL84] R. J. Brachman, H. J. Levesque. "The tractability of subsumption in frame based description languages." *Proceedings of the 4th National Conference of the AAAI,* pp. 34–37, Austin, Tex., 1984.

[BS85] R. J. Brachman, J. Schmolze, "An overview of the KL-ONE knowledge representation system." *Cognitive Science*, 9 (2), pp. 171-216, 1985.

[DLNN91] F.M. Donini, M. Lenzerini, D. Nardi, W. Nutt. "The complexity of concept languages." To appear in the *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, Boston, 1991.

[GJ79] M. R. Garey, D. S. Johnson. *Computers and intractability— A guide to the theory of NP-completeness.* Freeman, San Francisco, Cal., 1979.

[HNS90] B. Hollunder, W. Nutt, M. Schmidt-Schauß. "Subsumption algorithms for concept description languages." *Proceedings of the 9th ECAI,* Pitman, London, 1990.

[LB87] H. J. Levesque, R. J. Brachman. "Expressiveness and tractability in knowledge representation and reasoning." *Computational Intelligence*, **3**:78–93, 1987.

[Neb88] B. Nebel. "Computational complexity of terminological reasoning in BACK." *Artificial Intelligence*, **34**(3):371–383, 1988.

[Neb90] B. Nebel. *Reasoning and revision in hybrid representation systems*, Lecture Notes in Artificial Intelligence 422, Springer Verlag, 1990.

[Neb90] B. Nebel. "Terminological reasoning is inherently intractable." *Artificial Intelligence*, **43**:235–249, 1990.

[NS90] B. Nebel, G. Smolka. "Representation and reasoning with attributive descriptions." In K.H. Bläsius, U.Hedtstück, C.-R. Rollinger (editors), *Sorts and types in artificial intelligence*, Lecture Notes in Artificial Intelligence 418, Springer Verlag, 1990.

[SS88] M. Schmidt-Schauß, G. Smolka. "Attributive concept descriptions with complements." SEKI Report SR-88-21, FB Informatik, Universität Kaiserslautern, D-6750 Kaiserslautern, Germany, 1988.

[SS91] M. Schmidt-Schauß, G. Smolka. "Attributive concept descriptions with complements." To appear in *Artificial Intelligence*, **47**, 1991.

[Smu68] R. M. Smullyan. *First-order logic*. Springer Verlag, Berlin 1968.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
Stuhlsatzenhausweg 3
6600 Saarbrücken 11
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen oder die aktuelle Liste von erhältlichen Publikationen können bezogen werden von der oben angegebenen Adresse.

DFKI Publications

The following DFKI publications or the list of currently available publications can be ordered from the above address.

DFKI Research Reports

RR-90-01
Franz Baader
Terminological Cycles in KL-ONE-based Knowledge Representation Languages
33 pages

RR-90-02
Hans-Jürgen Bürckert
A Resolution Principle for Clauses with Constraints
25 pages

RR-90-03
Andreas Dengel & Nelson M. Mattos
Integration of Document Representation, Processing and Management
18 pages

RR-90-04
Bernhard Hollunder & Werner Nutt
Subsumption Algorithms for Concept Languages
34 pages

RR-90-05
Franz Baader
A Formal Definition for the Expressive Power of Knowledge Representation Languages
22 pages

RR-90-06
Bernhard Hollunder
Hybrid Inferences in KL-ONE-based Knowledge Representation Systems
21 pages

RR-90-07
Elisabeth André, Thomas Rist
Wissensbasierte Informationspräsentation:
Zwei Beiträge zum Fachgespräch Graphik und KI:
1. Ein planbasierter Ansatz zur Synthese illustrierter Dokumente
2. Wissensbasierte Perspektivenwahl für die automatische Erzeugung von 3D-Objektdarstellungen

24 pages

RR-90-08
Andreas Dengel
A Step Towards Understanding Paper Documents
25 pages

RR-90-09
Susanne Biundo
Plan Generation Using a Method of Deductive Program Synthesis
17 pages

RR-90-10
Franz Baader, Hans-Jürgen Bürckert, Bernhard Hollunder, Werner Nutt, Jörg H. Siekmann
Concept Logics
26 pages

RR-90-11
Elisabeth André, Thomas Rist
Towards a Plan-Based Synthesis of Illustrated Documents
14 pages

RR-90-12
Harold Boley
Declarative Operations on Nets
43 pages

RR-90-13
Franz Baader
Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles
40 pages

RR-90-14
Franz Schmalhofer, Otto Kühn, Gabriele Schmidt
Integrated Knowledge Acquisition from Text, Previously Solved Cases, and Expert Memories
20 pages

RR-90-15
Harald Trost
The Application of Two-level Morphology to Non-concatenative German Morphology
13 pages

RR-90-16
Franz Baader, Werner Nutt
Adding Homomorphisms to Commutative/Monoidal Theories, or: How Algebra Can Help in Equational Unification
25 pages

RR-91-01
Franz Baader, Hans-Jürgen Bürckert, Bernhard Nebel, Werner Nutt, and Gert Smolka
On the Expressivity of Feature Logics with Negation, Functional Uncertainty, and Sort Equations
20 pages

RR-91-02
Francesco Donini, Bernhard Hollunder, Maurizio Lenzerini, Alberto Marchetti Spaccamela, Daniele Nardi, Werner Nutt
The Complexity of Existential Quantification in Concept Languages
22 pages

RR-91-03
B.Hollunder, Franz Baader
Qualifying Number Restrictions in Concept Languages
20 pages

DFKI Technical Memos

TM-89-01
Susan Holbach-Weber
Connectionist Models and Figurative Speech
27 pages

TM-90-01
Som Bandyopadhyay
Towards an Understanding of Coherence in Multimodal Discourse
18 pages

TM-90-02
Jay C. Weber
The Myth of Domain-Independent Persistence
18 pages

TM-90-03
Franz Baader, Bernhard Hollunder
KRIS: Knowledge Representation and Inference System
-System Description-
15 pages

TM-90-04
Franz Baader, Hans-Jürgen Bürckert, Jochen Heinsohn, Bernhard Hollunder, Jürgen Müller, Bernhard Nebel, Werner Nutt, Hans-Jürgen Profitlich
Terminological Knowledge Representation: A Proposal for a Terminological Logic
7 pages

DFKI Documents

D-89-01
Michael H. Malburg & Rainer Bleisinger
HYPERBIS: ein betriebliches Hypermedia-Informationssystem
43 Seiten

D-90-01
DFKI Wissenschaftlich-Technischer Jahresbericht 1989
45 pages

D-90-02
Georg Seul
Logisches Programmieren mit Feature -Typen
107 Seiten

D-90-03
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner
Abschlußbericht des Arbeitspaketes PROD
36 Seiten

D-90-04
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner
STEP: Überblick über eine zukünftige Schnittstelle zum Produktdatenaustausch
69 Seiten

D-90-05
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner
Formalismus zur Repräsentation von Geo-metrie- und Technologieinformationen als Teil eines Wissensbasierten Produktmodells
66 Seiten

D-90-06
Andreas Becker
The Window Tool Kit
66 Seiten

The Complexity of Existential Quantification in Concept Languages

Francesco Donini, Bernhard Hollunder, Maurizio Lenzerini,
Alberto Marchetti Spaccamela, Daniele Nardi, Werner Nutt

RR-91-02
Research Report