Attributive concept descriptions with complements

Manfred Schmidt-Schauß
Software AG, Uhlandstr. 12, 6100 Darmstadt, FRG

Gert Smolka
DFKI and Universität des Saarlandes, Stuhlsatzenhausweg 3, 6600 Saarbrücken 11, FRG

Received: February 1989
Revised: January 1990

Abstract

Schmidt-Schauß, M. and G. Smolka, Attributive concept descriptions with complements, Artificial Intelligence 48 (1991) 1–26.

We investigate the consequences of adding unions and complements to attributive concept descriptions employed in terminological knowledge representation languages. It is shown that deciding coherence and subsumption of such descriptions are PSPACE-complete problems that can be decided with linear space.

1. Introduction

Research in knowledge representation has led to the development of so-called terminological representation languages [4, 6, 14, 22, 24, 26–28, 35], which originated with Brachman's KL-ONE [7] and grew out of research in semantic networks and frame systems. These languages are based on attributive concept descriptions providing for the partial description of objects by means of stating membership in concepts and giving restrictions for attributes called roles. Brachman and Levesque's [5, 20, 21] Tarski-style semantics for attributive concept descriptions models concepts as sets and roles as binary relations. The denotation of an attributive concept description is then a set obtained inductively from the description and the denotations of the occurring concepts and roles.

The distinctive formation rules for concept descriptions are role quantifications of the form $\forall R\colon C$ and $\exists R\colon C$, where R is a role and C is a concept description. A universal role quantification $\forall R\colon C$ can be read as "all objects for which all R are in C", and an existential role quantification $\exists R\colon C$ can be read as "all objects for which there is an R in C". Given an interpretation $\mathcal{I}$ for the occurring concept and role symbols, these descriptions denote the sets

$$(\forall R\colon C)^{\mathcal{I}} = \{a \in D^{\mathcal{I}} \mid \forall (a, b) \in R^{\mathcal{I}}\colon b \in C^{\mathcal{I}}\}$$

and

$$(\exists R\colon C)^{\mathcal{I}} = \{a \in D^{\mathcal{I}} \mid \exists (a, b) \in R^{\mathcal{I}}\colon b \in C^{\mathcal{I}}\}\,,$$

where $D^{\mathcal{I}}$ is the domain of $\mathcal{I}$. Since concept descriptions denote sets, it's clear how to provide for concept intersection (written $C \sqcap D$), concept union (written $C \sqcup D$), and concept complement (written $\neg C$). Furthermore, we assume that there is a concept "top" (written $\top$) that denotes the entire domain $D^{\mathcal{I}}$ of the interpretation $\mathcal{I}$.

Concept descriptions can be used to state necessary or defining conditions for concepts by means of so-called terminological axioms, which are either inclusions or equations between concepts and concept descriptions. For instance, we may write the terminological axioms

$$\mathsf{man} \doteq \mathsf{person} \sqcap \mathsf{sex}\colon \mathsf{male}\,,$$

$$\mathsf{parent} \doteq \mathsf{person} \sqcap \exists \mathsf{child}\colon \top\,,$$

$$\mathsf{father} \doteq \mathsf{parent} \sqcap \mathsf{man}\,,$$

$$\mathsf{grandfather} \doteq \mathsf{father} \sqcap \exists \mathsf{child}\colon \mathsf{parent}\,,$$

to define the concepts man, parent, father and grandfather in terms of the concepts person and male and the roles sex and child. With complements we can express relations like

$$\mathsf{woman} \doteq \mathsf{person} - \mathsf{man}\,,$$

where the concept difference $\mathsf{person} - \mathsf{man}$ stands for $\mathsf{person} \sqcap \neg \mathsf{man}$. This axiom implies that man and woman are disjoint and that every person is either a woman or a man.

We can also express conditional knowledge like "animals that are featherless bipeds are humans":

$$\mathsf{animal} \sqsubseteq \mathsf{featherless\_biped} \rightarrow \mathsf{human}\,.$$

The symbol "$\sqsubseteq$" stands for set inclusion and the concept implication

$$\mathsf{featherless\_biped} \rightarrow \mathsf{human}$$

is an abbreviation for

$$\neg\text{featherless\_biped} \sqcup \text{human}\,,$$

which is equivalent to

$$\neg(\text{featherless\_biped} \sqcap \neg\text{human})\,.$$

Given a set T of terminological axioms, a concept description C is *coherent* in T if there exists at least one model of T in which C denotes a nonempty set. Furthermore, a concept description C *subsumes* a concept description D in T if C denotes in every model of T a superset of D. Checking coherence and subsumption belong to the basic reasoning services of terminological representation systems. Under certain restrictions on the terminological axioms T [24–26], which are usually enforced in existing systems, coherence and subsumption in T can be reduced to coherence and subsumption with respect to all possible interpretations. If complements are available, subsumption and incoherence are linear-time reducible to each other since

$$C \text{ subsumes } D \leftrightarrow \neg C \square D \text{ incoherent}\,,$$
$$C \text{ incoherent} \leftrightarrow \neg\top \text{ subsumes } C\,.$$

Brachman and Levesque [5, 20, 21] show that the subsumption problem for descriptions built with concepts, intersections, universal role quantifications and unqualified existential role quantifications ($\exists R\colon \top$) can be solved in quadratic time while it is co-NP-hard for a, seemingly, slightly more expressive language. Nebel [23] shows that the subsumption problem for the concept descriptions employed in the terminological representation system BACK is also co-NP-hard. Schmidt-Schauß [31] shows that subsumption is undecidable for descriptions with so-called role value maps, which are available in KL-ONE [7] and NIKL [14]. These results are all obtained under the assumption that there are no terminological axioms. Recently, Nebel [25] has shown that even for the most simple attributive concept descriptions subsumption with respect to terminological axioms is co-NP-hard.

In this paper we investigate the computational complexity of subsumption and coherence of attributive concept descriptions built from concepts, intersections, complements and universal role quantifications. This attributive concept description language, called $\mathcal{ALC}$, is fairly expressive and enjoys pleasant mathematical properties. $\mathcal{ALC}$ is propositionally complete in that all boolean set operations can be expressed. Union and difference, for instance, can be expressed with

$$C \sqcup D \equiv \neg(\neg C \sqcap \neg D)\,,$$
$$C - D \equiv C \sqcap \neg D\,.$$

Furthermore, existential role quantification can be expressed with

$$\exists R\colon C \equiv \neg(\forall R\colon \neg C)\,.$$

As pointed out before, subsumption in $\mathcal{ALC}$ is linear-time equivalent to incoherence.

We show that deciding coherence and subsumption of $\mathcal{ALC}$ descriptions are PSPACE-complete problems and give a linear-space exponential-time algorithm for deciding coherence of $\mathcal{ALC}$ descriptions (everything with respect to all interpretations). Furthermore, we prove more specific complexity results for sublanguages of $\mathcal{ALC}$. The focus of this paper is on coherence checking, which is a basic operation of the assertional reasoner in hybrid terminological representation systems [24, 26]. In this respect our paper complements the existing literature on terminological reasoning that concentrates on subsumption checking. Although in the case of $\mathcal{ALC}$ coherence and subsumption checking reduce to each other in linear time, starting with coherence checking leads to technically simpler algorithms and proofs.

Although the expressivity gained by complements could be very useful for practical applications [10], attributive concept descriptions with complements have not been studied theoretically before nor are they present in implemented systems. One of the reasons may be Brachman and Levesque's [5, 20, 21] influential argument that a knowledge representation service should be designed such that it can be implemented with polynomial complexity, and their suggestion that expressive power should be traded for tractability. Nebel's [25] intractability result shows that this tradeoff is not available for terminological reasoning systems: subsumption with respect to terminological axioms is intractable even for the weakest attributive concept descriptions.

The conflict between expressive power and computational tractability has led to the use of incomplete algorithms in existing theorem proving and knowledge representation systems. The usefulness of such systems certainly increases if one has a good understanding of the sources of incompleteness. We believe that the study of complete methods contributes to this understanding and is essential for the development of better, and better described, partial methods.

The paper is organized as follows. Section 2 introduces attributive concept descriptions formally and states their obvious properties. Section 3 shows how coherence of concept descriptions can be expressed in terms of constraint systems, which will serve as the base for our computational investigations. Section 4 gives completion rules for constraint systems that yield a proof-theoretic characterization of coherence. In Section 5 we obtain coherence checking algorithms for $\mathcal{ALC}$ and some of its sublanguages by organizing the completion rules with suitable control strategies. In Section 6 we establish the PSPACE-hardness of coherence checking in $\mathcal{ALC}$ by reducing the validity problem for quantified boolean formulas to it. Section 7 relates our results with work on feature descriptions used in unification grammar formalisms and logic programming.

2. Attributive concept descriptions

Let two disjoint alphabets of symbols, called *concepts* and *roles*, respectively, be given. We assume that $\top$ (read "top") is a concept symbol. The letters A and B will always denote concept symbols and the letter R will always denote a role symbol.

The attributive concept descriptions of the language $\mathcal{ALC}$ are given by the abstract syntax rule:

$$C, D \rightarrow A \mid \forall R\colon C \mid \exists R\colon C \mid C \sqcap D \mid C \sqcup D \mid \neg C\,.$$

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

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

A concept description C is *coherent* if there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}}$ is nonempty. A concept description C is *subsumed* by a concept description D if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every interpretation $\mathcal{I}$. A concept description C is *equivalent* to a concept description D (written $C \sim D$) if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every interpretation $\mathcal{I}$.

Proposition 2.1. *A concept description C is subsumed by a concept description D if and only if the concept description $C \sqcap \neg D$ is incoherent.*

Hence a coherence checking algorithm for $\mathcal{ALC}$ can also be used for testing subsumption in $\mathcal{ALC}$. Since a concept description is incoherent if and only if it is subsumed by $\neg\top$ (the empty set), a subsumption checking algorithm can also be used for coherence checking.

The syntax of $\mathscr{ALC}$ is redundant. For instance, $\top$ is equivalent to $A \sqcup \neg A$ for every concept symbol A, $\exists R{:}\ C$ is equivalent to $\neg(\forall R{:}\ \neg C)$ and $C \sqcup D$ is equivalent to $\neg(\neg C \sqcap \neg D)$.

The redundant syntax provides for the simplification of complex complements to *simple complements* of the form $\neg A$, where A is a concept symbol. This can be done in linear time by pushing down complex complements with the following equivalences:

$$\neg(\forall R{:}\ C) \sim \exists R{:}\ \neg C\,,$$

$$\neg(\exists R{:}\ C) \sim \forall R{:}\ \neg C\,,$$

$$\neg(C \sqcap D) \sim \neg C \sqcup \neg D\,,$$

$$\neg(C \sqcup D) \sim \neg C \sqcap \neg D\,,$$

$$\neg\neg C \sim C\,.$$

We call a concept description *simple* if it contains only simple complements.

Proposition 2.2. *For every concept description one can compute in linear time an equivalent simple concept description.*

We define three sublanguages of $\mathscr{ALC}$:

- $\mathscr{ALE}$ is obtained from $\mathscr{ALC}$ by allowing only simple concept descriptions containing no unions;
- $\mathscr{ALU}$ is obtained from $\mathscr{ALC}$ by allowing only simple concept descriptions and restricting existential role quantifications to the form $\exists R{:}\ \top$;
- $\mathscr{AL}$ is obtained from $\mathscr{ALC}$ by allowing only simple concept descriptions containing no unions and restricting existential role quantifications to the form $\exists R{:}\ \top$.

Note that $\mathscr{AL}$ is the intersection of $\mathscr{ALE}$ and $\mathscr{ALU}$. The names of these languages are cooked up as follows: $\mathscr{ALU}$ is obtained from $\mathscr{AL}$ by adding unions, $\mathscr{ALE}$ is obtained from $\mathscr{AL}$ by adding general existential role quantifications, and $\mathscr{ALC}$ is obtained from $\mathscr{AL}$ by adding general complements. We consider $\mathscr{AL}$ to be the minimal sensible attributive concept description language. Without simple complements no two concept descriptions could be disjoint.

Proposition 2.3. *Deciding coherence of $\mathscr{ALU}$ concept descriptions is NP-hard, and deciding subsumption of $\mathscr{ALU}$ concept descriptions is co-NP-hard. This holds already for descriptions not containing roles.*

Proof. Since C is incoherent if and only if C is subsumed by $\neg\top$, it suffices to show that checking coherence is NP-hard.

It is well-known that deciding the satisfiability of propositional formulas in conjunctive normal form (CNF) is an NP-complete problem (see, for instance, [12]). A propositional formula in CNF can be seen as an $\mathcal{ALU}$ concept description by regarding propositional variables as concepts, conjunctions as intersections, disjunctions as unions, and negations as complements.

Let $F = F_1 \wedge \cdots \wedge F_n$ be a propositional formula in CNF, where every F_i is a disjunction of literals. Obviously, F is satisfiable if and only if one can choose in every F_i a literal L_i such that $L_1, \ldots, L_n$ don't contain a complementary pair.

Suppose F is satisfiable. Then there exist $L_1, \ldots, L_n$ as specified above. Let $\mathcal{I}$ be the interpretation such that

$$D^{\mathcal{I}} = \{1\}\,,$$

$$A^{\mathcal{I}} = \begin{cases} \{1\}, & \text{if } A = L_i \text{ for some } i, \\ \emptyset, & \text{otherwise,} \end{cases}$$

$$R^{\mathcal{I}} = \emptyset \quad \text{for every role } R\,.$$

Then $F^{\mathcal{I}} = L_i^{\mathcal{I}} = \{1\}$ for $i \in 1 \,..\, n$, which shows that F is a coherent concept description.

Suppose F is a coherent concept description. Then there exists an interpretation $\mathcal{I}$ and an $a \in D^{\mathcal{I}}$ such that $a \in F^{\mathcal{I}}$. Hence every F_i contains a literal L_i such that $a \in L_i^{\mathcal{I}}$. Thus $L_1, \ldots, L_n$ don't contain a complementary pair, which shows that F is satisfiable. □

In this paper we are going to prove the following results:

(1) Checking coherence and subsumption of $\mathcal{ALC}$ concept descriptions are PSPACE-complete problems that can be decided with linear space.
(2) Incoherence (not coherence) of $\mathcal{ALE}$ concept descriptions can be decided in nondeterministic linear time.
(3) Checking coherence of $\mathcal{ALU}$ concept descriptions is an NP-complete problem (we have already shown that checking subsumption of $\mathcal{ALU}$ concept descriptions is a co-NP-hard problem).
(4) Coherence of $\mathcal{AL}$ concept descriptions can be checked in linear time.

Using some of the techniques of this paper, Donini et al. [8] recently showed that checking coherence of $\mathcal{ALE}$ concept descriptions is co-NP-complete, and that checking subsumption of $\mathcal{ALE}$ concept descriptions is NP-complete. Hollunder [13] shows that $\mathcal{ALC}$ extended with number constraints or role hierarchies still has a PSPACE-complete subsumption relation. Furthermore, he shows that $\mathcal{ALC}$ remains decidable if both extensions are added.

The relationship between our attributive concept description languages and the languages $\mathcal{FL}$ and $\mathcal{FL}^-$ in [5, 20, 21] is as follows:

$$\begin{aligned}
\mathcal{AL} &\equiv \mathcal{FL}^{-} + \text{simple complements}\,,\\
\mathcal{ALC} &\equiv \mathcal{FL}^{-} + \text{general complements}\\
&\equiv \mathcal{FL} + \bot + \mathit{SELF}\,,
\end{aligned}$$

where $\bot$ is a concept and SELF is a role whose interpretations are fixed as

$$\begin{aligned}
&\bot^{\mathcal{I}} = \emptyset\,,\\
&\mathit{SELF}^{\mathcal{I}} = \{(a, a) \mid a \in D^{\mathcal{I}}\}
\end{aligned}$$

in every interpretation $\mathcal{I}$. The first equation is obvious. The other two equations follow from the equivalences:

$$\begin{aligned}
&(\textsc{restrict}\ (\textsc{restrict}\ R\ C)\ D) \sim (\textsc{restrict}\ R\ (\textsc{and}\ C\ D))\,,\\
&(\textsc{all}\ (\textsc{restrict}\ R\ C)\ D) \sim \forall R{:}\,(\neg C \sqcup D)\,,\\
&(\textsc{some}\ (\textsc{restrict}\ R\ C)) \sim \exists R{:}\ C\,,\\
&(\textsc{all}\ (\textsc{restrict}\ \mathit{SELF}\ C) \bot) \sim \neg C\,,\\
&\forall \mathit{SELF}{:}\ C \sim C\,,\\
&\exists \mathit{SELF}{:}\ C \sim C\,.
\end{aligned}$$

Since every $\mathcal{FL}$-concept description is coherent (see Theorem 4.6), $\mathcal{FL}^{-}$ is a proper sublanguage of $\mathcal{AL}$ and $\mathcal{FL}$ is a proper sublanguage of $\mathcal{ALC}$.

3. Constraint systems

The applicative structure of concept descriptions is rather unsuitable for devising coherence checking algorithms. However, every concept description can be translated in linear time into a constraint system such that the concept description is coherent if and only if the constraint system is satisfiable. For constraint systems we will give simplification rules that preserve satisfiability and unsatisfiability. With an appropriate control these simplification rules yield transparent satisfiability checking algorithms.

Every $\mathcal{ALC}$ description can be translated into a formula of predicate logic such that the concept description is coherent if and only if the formula is satisfiable. This translation translates role quantifiers into the corresponding quantifiers of predicate logic. The constraints used here are not standard formulas of predicate logic but are specially designed for $\mathcal{ALC}$.

We assume the existence of two further disjoint alphabets of symbols, called *individual* and *concept variables*, respectively. The letters x, y, z will always range over individual variables and the letters X, Y, Z will always range over concept variables.

Let $\mathcal{I}$ be an interpretation. An *$\mathcal{I}$-assignment* is a function α that maps every individual variable to an element of $D^{\mathcal{I}}$ and every concept variable to a subset of $D^{\mathcal{I}}$. We use $\text{ASS}^{\mathcal{I}}$ to denote the set of all $\mathcal{I}$-assignments.

A *constraint* is a piece of abstract syntax having one of the forms:

$$\begin{array}{lll} X \sqsubseteq C\,, & X(\forall R)Y\,, & X(\exists R)Y\,, \\ X \sqsubseteq Y \sqcup Z\,, & x\colon X\,, & xRy\,, \end{array}$$

where C in the first form must be a simple concept description. Given an interpretation $\mathcal{I}$, we extend the interpretation function $\cdot^{\mathcal{I}}$ to constraints by interpreting them as sets of $\mathcal{I}$-assignments:

$$\begin{aligned} &(X \sqsubseteq C)^{\mathcal{I}} = \{\alpha \in \text{ASS}^{\mathcal{I}} \mid \alpha(X) \subseteq C^{\mathcal{I}}\}\,, \\ &(X(\forall R)Y)^{\mathcal{I}} = \{\alpha \in \text{ASS}^{\mathcal{I}} \mid \forall a \in \alpha(X) \forall (a, b) \in R^{\mathcal{I}} \colon b \in \alpha(Y)\}\,, \\ &(X(\exists R)Y)^{\mathcal{I}} = \{\alpha \in \text{ASS}^{\mathcal{I}} \mid \forall a \in \alpha(X) \exists (a, b) \in R^{\mathcal{I}} \colon b \in \alpha(Y)\}\,, \\ &(X \sqsubseteq Y \sqcup Z)^{\mathcal{I}} = \{\alpha \in \text{ASS}^{\mathcal{I}} \mid \alpha(X) \subseteq \alpha(Y) \cup \alpha(Z)\}\,, \\ &(x\colon X)^{\mathcal{I}} = \{\alpha \in \text{ASS}^{\mathcal{I}} \mid \alpha(x) \in \alpha(X)\}\,, \\ &(xRy)^{\mathcal{I}} = \{\alpha \in \text{ASS}^{\mathcal{I}} \mid (\alpha(x), \alpha(y)) \in R^{\mathcal{I}}\}\,. \end{aligned}$$

An assignment in $\phi^{\mathcal{I}}$ is called a *solution* of the constraint ϕ in $\mathcal{I}$.

A *constraint system* is a finite, nonempty set of constraints. Given an interpretation $\mathcal{I}$, the *solutions* of a constraint system S in $\mathcal{I}$ are defined as follows:

$$S^{\mathcal{I}} = \bigcap_{\phi \in S} \phi^{\mathcal{I}}\,.$$

A constraint system S is called *satisfiable* if there exists an interpretation $\mathcal{I}$ in which S has a solution. The next proposition gives a translation of simple $\mathcal{ALC}$ concept descriptions into constraint systems such that coherence corresponds to satisfiability:

Proposition 3.1. *Let x be an individual variable and X be a concept variable. Then a simple concept description C is coherent if and only if the constraint system $\{x \in X, X \sqsubseteq C\}$ is satisfiable.*

A constraint system S is *simple* if for every constraint $X \sqsubseteq C$ in S the concept description C is either a concept symbol different from $\top$ or a complemented concept symbol.

The following *unfolding rules* can be used to simplify general constraint systems to simple constraint systems:

$$X \sqsubseteq \forall R{:}\, C \to X(\forall R)Y,\ Y \sqsubseteq C,$$
where Y is a new variable,

$$X \sqsubseteq \exists R{:}\, C \to X(\exists R)Y,\ Y \sqsubseteq C,$$
where Y is a new variable,

$$X \sqsubseteq C \sqcap D \to X \sqsubseteq C,\ X \sqsubseteq D$$

$$X \sqsubseteq C \sqcup D \to X \sqsubseteq Y \sqcup Z,\ Y \sqsubseteq C,\ Z \sqsubseteq D,$$
where Y, Z are new variables,

$$X \sqsubseteq \top \to \text{nothing}.$$

Proposition 3.2. *Let a constraint system S' be obtained from a constraint system S by the application of an unfolding rule. Then S is satisfiable if and only if S' is satisfiable.*

Proposition 3.3. *For every constraint system S one can compute in linear time a simple constraint system S' such that S is satisfiable if and only if S' is satisfiable.*

A simple constraint system defines a labeled, directed graph, called its *skeleton*, as follows: every concept variable occurring in the constraint system is taken as a node, the constraints $X(\exists R)Y$ and $X(\forall R)Y$ define existential and universal edges from X to Y, respectively, and a constraint $X \sqsubseteq Y \sqcup Z$ defines an or-connected pair of edges from X to Y and X to Z. Furthermore, the constraints $X \sqsubseteq A$ and $X \sqsubseteq \neg A$ define A and $\neg A$, respectively, as labels of the node X. Thus every node has a finite, possibly empty set of labels, where every label is either a concept symbol different from $\top$ or a complemented concept symbol. The individual constraints $x{:}\, X$ and xRy don't contribute to the skeleton.

A *constraint tree* (*constraint forest*) is a simple constraint system whose skeleton is a tree (forest). A constraint tree T is *fresh* if it can be obtained by unfolding a simple $\mathcal{ALC}$ concept description. Note that a fresh constraint tree has only one constraint containing an individual variable, which has the form $x{:}\, X$, where X is the root of the tree. Now we can formulate the main result of this section:

Theorem 3.4. *For every concept description C one can compute in linear time a fresh constraint tree T such that C is coherent if and only if T is satisfiable.*

Proof. First C is transformed into a simple concept description using the simplification rules given in the previous section. Then the corresponding constraint system is created, which is then simplified to a fresh constraint tree using the unfolding rules. All three steps require at most linear time and preserve coherence/satisfiability and incoherence/unsatisfiability. □

4. Pebble semantics

We now define so-called complete constraint systems whose satisfiability can be checked in linear time. We will show that every fresh constraint tree can be transformed into a complete constraint system using completion rules that preserve satisfiability in both directions. This yields a proof-theoretic characterization of the coherence of $\mathcal{ALC}$ descriptions called pebble semantics. Pebble semantics provides a framework in which coherence checking algorithms are obtained as completion algorithms for fresh constraint trees.

A *clash* is a constraint system having either the form $\{x\colon X,\ X \sqsubseteq \neg\top\}$ or the form $\{x\colon X,\ X \sqsubseteq A,\ x\colon Y,\ Y \sqsubseteq \neg A\}$.

Proposition 4.1. *Every constraint system containing a clash is unsatisfiable. Furthermore, one can check in linear time whether a constraint system contains a clash.*

A complete constraint system is a simple constraint system S satisfying the following conditions:

(1) If $x\colon X$ and $X(\exists R)Y$ are in S, then there exists a variable y such that $y\colon Y$ and xRy are in S.
(2) *If* $x\colon X$, $X(\forall R)Y$ and xRy are in S, then $y\colon Y$ is in S.
(3) *If* $x\colon X$ and $X \sqsubseteq Y \sqcup Z$ are in S, then $x\colon Y$ or $x\colon Z$ is in S.

Proposition 4.2. *A complete constraint system is satisfiable if and only if it contains no clash.*

Proof. One direction is obvious. To see the other direction, let S be a complete constraint system containing no clash. We define an interpretation $\mathcal{I}$ by taking for $D^{\mathcal{I}}$ the set of all individual variables occurring in S, for $A^{\mathcal{I}}$ the set of all x such that $x\colon X$ and $X \sqsubseteq A$ are in S for some X, and by taking for $R^{\mathcal{I}}$ the set of all pairs (x, y) such that xRy is in S. Now we obtain a solution $\alpha \in S^{\mathcal{I}}$ by mapping individual variables to themselves and taking for $\alpha(X)$ the set of all x such that $x\colon X$ is in S. □

A fresh constraint tree can be completed by propagating downwards the individual variable belonging to its root. At existential edges new individual variables are introduced as needed. The authors liked to visualize the individual variables in this propagation process as pebbles. There are three completion rules whose logical properties are stated in the following proposition:

Proposition 4.3. *Let S be a constraint system. Then:*

(1) *if x: X and $X(\exists R)Y$ are in S and y is an individual variable not occurring in S, then S is satisfiable if and only if $S \cup \{xRy, y\colon Y\}$ is satisfiable;*
(2) *if x: X, xRy and $X(\forall R)Y$ are in S, then S is satisfiable if and only if $S \cup \{y\colon Y\}$ is satisfiable;*
(3) *if x: X and $X \sqsubseteq Y \sqcup Z$ are in S, then S is satisfiable if and only if $S \cup \{x\colon Y\}$ or $S \cup \{x\colon Z\}$ is satisfiable.*

To obtain an algorithm, we need to impose some control that ensures that after finitely many completion steps no further completion step is applicable. This leads to the completion rules given in Fig. 1.

Proposition 4.4. *The basic completion rules in Fig.* 1 *have the following properties:*

(1) *There is no infinite chain of completion steps issuing from a fresh constraint tree.*
(2) *A simple constraint system is complete if and only if none of the completion rules applies to it.*
(3) *If T' is obtained from a constraint tree T by one of the completion rules, then T' is a constraint tree and T is satisfiable if T' is satisfiable.*

A *completion* of a constraint tree T is a complete constraint tree that can be obtained from T by finitely many applications of the basic completion rules in Fig. 1.

Next we state the soundness and completeness of our pebble semantics for fresh constraint trees. Soundness means that a fresh constraint tree is satisfiable if it has a clash-free completion. Completeness means that every satisfiable fresh constraint tree has a clash-free completion.

(1) $S \rightarrow_{\exists} \{y\colon Y, xRy\} \cup S$
 if x: X and $X(\exists R)Y$ are in S, there exists
 no variable z such that xRz and z: Y are in S,
 and y is a variable not occurring in S

(2) $S \rightarrow_{\forall} \{y\colon Y\} \cup S$
 if x: X, xRy and $X(\forall R)Y$ are in S and y: Y is not in S

(3) $S \rightarrow_{\sqcup} \{x\colon Z\} \cup S$
 if x: X and $X \sqsubseteq Y_1 \sqcup Y_2$ are in S,
 neither x: Y_1 or x: Y_2 is in S, and Z is either Y_1 or Y_2

Fig. 1. The basic completion rules for constraint trees.

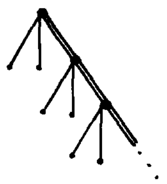

Fig. 2. A family of skeletons for which the number of individual variables in every completion is exponential in the size of the skeleton. These skeletons can be obtained from the concept descriptions $(\exists R\colon \top)\sqcap(\exists R\colon \top)\sqcap(\forall R\colon \cdots)$. A "double-line edge" represents a universal edge and a "single-line edge" represents an existential edge. Note that only one role symbol is used.

Theorem 4.5. *Every fresh constraint tree has a completion. Furthermore, a fresh constraint tree is satisfiable if and only if it has a clash-free completion.*

Proof. Follows from the preceding propositions. □

Thus we have an algorithm for deciding coherence and subsumption of $\mathcal{ALC}$ concept descriptions. The example in Fig. 2 shows that the completions of a fresh constraint tree can all be exponentially larger than the initial tree. However, there is no need to keep the entire completion in memory. In the next section we will give a smarter control for the completion rules yielding an algorithm that requires only linear space.

Theorem 4.6. *Every $\mathcal{FL}$ concept description is coherent.*

Proof. Let F be an $\mathcal{FL}$ concept description and T be a simple constraint tree obtained by unfolding F's translation into an $\mathcal{ALC}$ concept description. Although F contains neither unions nor complements, T does since they are introduced by the translation rule

$$(\text{ALL (RESTRICT } R\ C)\ D) \rightarrow \forall R\colon (\neg C \sqcup D)\,.$$

However, if we complete T such that individual variables are always propagated to the right-hand sides of unions, we obtain a clash-free completion since then a pair $\{x\colon X,\ X\colon \neg A\}$ cannot occur. □

5. Upper complexity bounds

In this section we will prove upper complexity bounds for the attributive concept description languages $\mathcal{AL}$, $\mathcal{ALE}$, $\mathcal{ALU}$ and $\mathcal{ALC}$. In particular, we will show that coherence of $\mathcal{ALC}$ concept descriptions can be decided with linear space. The basic idea behind the linear-space algorithm is that a completion can be sliced up into so-called traces such that the completion contains a clash if and only if one of its traces contains a clash. While the size of completions can be exponential, the size of traces is linear in the size of the initial concept description. The algorithm systematically enumerates traces until it either finds a clash or has verified the existence of a clash-free completion.

A *partial completion* of a fresh constraint tree T is a constraint tree that can be obtained from T by finitely many applications of the completion rules. In particular, a fresh constraint tree is a partial completion of itself. Given a constraint tree T and individual variables x and y occurring in T, y is called a *successor* of x and x is called a *predecessor* of y if T contains a constraint xRy. An individual variable x occurring in a constraint tree T is called an *individual root* of T if T contains a constraint x: X such that X is the root of the skeleton of T.

Proposition 5.1. *Let T be a partial completion of a fresh constraint tree. Then T has a unique individual root, the individual root has no predecessor, and every other individual variable occurring in T has a unique predecessor.*

A partial completion U of a fresh constraint tree T is called a *trace* of T if

(1) no individual variable occurring in U has more than one successor;
(2) the completion rules $\rightarrow_{\forall}$ and $\rightarrow_{\sqcup}$ don't apply to U;
(3) every application of the completion rule $\rightarrow_{\exists}$ to U yields a constraint tree containing an individual variable having two successors.

Traces can be computed using the following restriction of the existential completion rule $\rightarrow_{\exists}$:

$$\begin{aligned} S \rightarrow_{T\exists} &\ \{y\colon Y, xRy\} \cup S \\ &\text{if } x\colon X \text{ and } X(\exists R)Y \text{ are in } S\,, \\ &\text{there is no constraint } xR'y' \text{ in } S, \text{ and} \\ &y \text{ is a variable not occurring in } S\,. \end{aligned}$$

We define the binary relation $\rightarrow_T$ on simple constraint systems by

$$\rightarrow_T := \rightarrow_{T\exists} \cup \rightarrow_{\forall} \cup \rightarrow_{\sqcup}\,,$$

where $\rightarrow_{T\exists}$, $\rightarrow_{\forall}$ and $\rightarrow_{\sqcup}$ are the relations on simple constraint systems given by the corresponding completion rules. Note that the *traces* of a fresh constraint tree T are the $\rightarrow_T$-normal forms of T (that is, constraint trees U such that $T \rightarrow_T^* U$ and $U \rightarrow_T V$ for no constraint tree V).

```
$eval : nat \times constraint\ tree \to bool$

$eval(x, S) =$
  if $\{x: X, X \sqsubseteq A, x: Y, Y \sqsubseteq \neg A\} \subseteq S \vee \{x: X, X \sqsubseteq \neg \top\} \subseteq S$
    then $false$
    elsif $\{x: X, X \sqsubseteq Y \sqcup Z\} \subseteq S \wedge (x: Y) \not\subseteq S \wedge (x: Z) \not\subseteq S$
      then $eval(x, \{x: Y\} \cup S) \vee eval(x, \{x: Z\} \cup S)$
      else let $y = x + 1$ in
        $\forall \{x: X, X(\exists R)Y\} \subseteq S$:
          $eval(y, \{y: Y\} \cup \{y: Z' \mid \exists \{x: Z, Z(\forall R)Z'\} \subseteq S\} \cup S)$
```

Fig. 3. A functional linear-space satisfiability checking algorithm for fresh constraint trees. If T is a fresh constraint tree and x is the individual root of T, then $eval(x, T) = true$ if and only if T has a clash-free completion, that is, is satisfiable. For convenience, individual variables are assumed to be natural numbers.

Proposition 5.2. *Let T be a fresh constraint tree. Then*:

(1) *if* $T \to_T^* T'$ *and* T' *contains the constraints* $x: X$ *and* $y: X$, *then* $x = y$ (*that is, no concept variable in* T' *has more than one individual variable associated with it*);

(2) *the length of* $\to_T$*-derivations issuing from T is bound linearly in the size of T*;

(3) *every trace of T is contained in a completion of T*;

(4) *every completion* $\tilde{T}$ *of T can be obtained as the union of finitely many traces.*

The recursive function *eval* in Fig. 3 yields *true* if the fresh constraint tree given as its argument has a clash-free completion and *false* otherwise. The maximum recursion depth is the height of the given tree. The function *eval* can be executed such that, besides some control information, at most a trace of the given tree is kept in memory. Hence *eval* needs at most space linear in the size of the input tree. The total correctness of *eval* follows from the propositions we have stated so far. Thus we have:

Theorem 5.3. *Coherence and subsumption of* $\mathcal{ALC}$ *concept descriptions can be checked with linear space.*

Proposition 5.4. *Let T be a fresh constraint tree not containing union constraints. Then*:

(1) *all completions of T are equal up to consistent renaming of individual variables*;

(2) *T is satisfiable* $\Leftrightarrow$ *T has a clash-free completion*
$\Leftrightarrow$ *every completion of T is clash-free*
$\Leftrightarrow$ *no trace of T contains a clash.*

Theorem 5.5. *Incoherence of $\mathcal{ALE}$ concept descriptions can be decided in nondeterministic linear time. Thus the coherence problem for $\mathcal{ALE}$ concept descriptions is in co-NP.*

Proof. We have to show that unsatisfiability of fresh constraint trees containing no union constraints can be decided in nondeterministic linear time. Since such constraint trees are unsatisfiable if and only if they have a trace containing a clash, this follows from the fact that every trace can be obtained by a $\rightarrow_T$-derivation whose length is bound linearly by the size of the constraint tree it is issuing from. □

To establish our upper complexity bounds for $\mathcal{ALU}$ and $\mathcal{AL}$, we need yet another restriction of the completion rule $\rightarrow_\exists$:

$$\begin{aligned} S &\rightarrow_{\mathsf{T}\exists} \{y\colon Y, xRy\} \cup S \\ &\quad \text{if } x\colon X \text{ and } X(\exists R)Y \text{ are in } S\,, \\ &\quad \text{there is no constraint } xRy' \text{ in } S, \text{ and} \\ &\quad y \text{ is a variable not occurring in } S\,. \end{aligned}$$

Note that $\rightarrow_{T\exists} \subseteq \rightarrow_{\mathsf{T}\exists}$ since $\rightarrow_{T\exists}$ can be applied to at most one existential edge at every level while $\rightarrow_{\mathsf{T}\exists}$ can be applied to several existential edges if they are labeled with different relation symbols. We define the binary relation $\rightarrow_{\mathsf{T}}$ on simple constraint systems by

$$\rightarrow_{\mathsf{T}} := \rightarrow_{\mathsf{T}\exists} \cup \rightarrow_{\forall} \cup \rightarrow_{\sqcup}\,.$$

Note that $\rightarrow_T \subseteq \rightarrow_{\mathsf{T}}$ since $\rightarrow_{T\exists} \subseteq \rightarrow_{\mathsf{T}\exists}$.

Proposition 5.6. *The length of $\rightarrow_{\mathsf{T}}$-derivations issuing from fresh constraint trees is bound linearly in the size of the initial tree.*

Proof. Let T be a fresh constraint tree and U be a constraint tree such that $T \rightarrow_{\mathsf{T}}^{*} U$. Then for every concept variable X in U there exists at most one individual variable x such that the constraint $x\colon X$ is in U. With this invariant the claim is obvious. □

The T-*completions* of a fresh constraint tree T are the $\rightarrow_{\mathsf{T}}$-normal forms of T.

Proposition 5.7. *Let T be a T-completion of a fresh constraint tree obtainable by unfolding an $\mathcal{ALU}$ concept description. If T contains the constraints $x\colon X$, $X(\exists R)Y$ and xRy, then T is satisfiable if and only if $T \cup \{y\colon Y\}$ is satisfiable.*

Proof. Follows from the fact that every concept variable reachable through an existential edge is an unlabeled leaf. □

Proposition 5.8. *Let T be a fresh constraint tree obtainable by unfolding an $\mathcal{ALU}$ concept description. Then T is satisfiable if and only if T has a clash-free $\top$-completion.*

Proof. If T is satisfiable, we know by Theorem 4.5 that T has a clash-free completion. Hence T has a clash-free $\top$-completion. To show the other direction, suppose T has a clash-free $\top$-completion U. Using the preceding proposition, we can extend U to a complete constraint tree V since concept variables reachable through existential edges are always leaves. Since U is clash-free and concept variables reachable through existential edges are not labeled, V is clash-free. Hence V is satisfiable. Since $T \subseteq U \subseteq V$, we thus know that T is satisfiable. □

Theorem 5.9. *Checking the coherence of $\mathcal{ALU}$ concept descriptions is an NP-complete problem.*

Proof. The NP-hardness was already stated in Proposition 2.3. That coherence checking is in NP follows from the preceding theorem since unfolding can be done in linear time, every $\top$-completion can be computed in nondeterministic linear time, and clash-freeness can be checked in linear time. □

Theorem 5.10. *Coherence of $\mathcal{AL}$ concept descriptions can be checked in linear time.*

Proof. Let T be a fresh constraint tree obtainable by unfolding an $\mathcal{AL}$ concept description. Then all $\top$-completions of T are equal up to consistent renaming of individual variables. Thus it suffices to compute any $\top$-completion and to check it for clashes. □

6. PSPACE-completeness

We now show that deciding coherence and subsumption of $\mathcal{ALC}$ concept descriptions are problems that are as hard as any problem that can be decided with polynomial space. Since we have proved in the last section that coherence and subsumption of $\mathcal{ALC}$ concept descriptions can be decided with linear space, we will be able to conclude that these problems are PSPACE-complete. The PSPACE-hardness of the coherence problem for $\mathcal{ALC}$ concept descriptions is proved by reducing the validity problem for quantified boolean formulas to it.

6.1. *Quantified boolean formulas*

We now review quantified boolean formulas whose validity problem (called QBF, for short) is known to be PSPACE-complete (see, for instance, [12]). We use a notation providing for a smooth reduction of QBF to the coherence problem for $\mathcal{ALC}$.

A *literal* is a nonzero integer. A *clause* is a nonempty sequence $l_1 \ldots l_n$ of literals such that $|l_1| \leq |l_2| \leq \cdots \leq |l_n|$. A *prefix* from m to n, where m and n are positive integers such that $m \leq n$, is a sequence

$$(Q_m m)(Q_{m+1} m+1) \ldots (Q_n n),$$

where each Q_i is either "$\forall$" or "$\exists$". A *quantified boolean formula* is a pair $P.M$, where, for some n, P is a prefix from 1 to n and M is a finite nonempty set of clauses containing only literals between $-n$ and n (M is called the *matrix* of the formula).

Let P be a prefix from m to n. A *P-assignment* is a mapping

$$\{m, m+1, \ldots, n\} \to \{0, 1\}.$$

An assignment α *satisfies* a literal l if $\alpha(l) = 1$ if l is positive and $\alpha(-l) = 0$ if l is negative. An assignment *satisfies* a clause if it satisfies at least one literal of the clause.

Let P be a prefix from m to n. A set A of P-assignments is *canonical* for P if it satisfies the following conditions:

(1) A is nonempty.
(2) If $P = (\exists m)P'$, then all assignments of A agree on m and, if P' is nonempty, $\{\alpha|_{\{m+1,\ldots,n\}} \mid \alpha \in A\}$ is canonical for P'.
(3) If $P = (\forall m)P'$, then
   (a) A contains an assignment that satisfies m and, if P' is nonempty, $\{\alpha|_{\{m+1,\ldots,n\}} \mid \alpha \in A \wedge \alpha(m) = 1\}$ is canonical for P';
   (b) A contains an assignment that satisfies $-m$ and, if P' is nonempty, $\{\alpha|_{\{m+1,\ldots,n\}} \mid \alpha \in A \wedge \alpha(m) = 0\}$ is canonical for P'.

A quantified boolean formula $P.M$ is *valid* if there exists a set A of P-assignments canonical for P such that every assignment in A satisfies every clause of M. An example of a valid quantified boolean formula written in a readable syntax is

$$\forall x \exists y.(\neg x \vee y) \wedge (x \vee \neg y).$$

The following theorem is taken from [12]:

Theorem 6.1. *Deciding the validity of quantified boolean formulas is a PSPACE-complete problem.*

6.2. *The reduction*

In the following we assume R to be a fixed role symbol and A to be a fixed concept symbol. Quantified boolean formulas are translated into $\mathcal{ALC}$ concept descriptions using the equation

$$[P.\{C_1, \ldots, C_n\}] = [P] \sqcap [C_1]^0 \sqcap \cdots \sqcap [C_n]^0 ,$$

where $[P]$ is defined inductively by the equations

$$\begin{aligned}
&[(\exists m)P] = ((\exists R\colon A) \square (\exists R\colon \neg A)) \square (\forall R\colon [P]) ,\\
&[(\forall m)P] = (\exists R\colon A) \square (\exists R\colon \neg A) \square (\forall R\colon [P]) ,\\
&[(\exists m)] = (\exists R\colon A) \square (\exists R\colon \neg A) ,\\
&[(\forall m)] = (\exists R\colon A) \square (\exists R\colon \neg A) ,
\end{aligned}$$

and $[C]^m$ is defined inductively by the equations

$$\begin{aligned}
&[lC]^m = \forall R\colon [lC]^{m+1} , \quad \text{if } |l| > m ,\\
&[mC]^m = A \sqcup [C]^m ,\\
&[-mC]^m = \neg A \sqcup [C]^m ,\\
&[l]^m = \forall R\colon [l]^{m+1} , \quad \text{if } |l| > m ,\\
&[m]^m = A ,\\
&[-m]^m = \neg A .
\end{aligned}$$

The number argument m of the translation function $[C]^m$ for clauses is needed to ensure that only unions of the form $A \sqcup C$ or $\neg A \sqcup C$ are introduced, which is essential. Figure 4 gives an example of a translation.

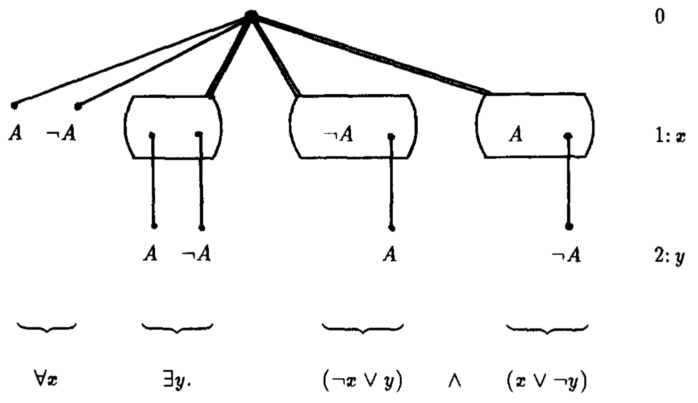

Fig. 4. A valid quantified boolean formula and its translation into a constraint tree.

To show that a quantified boolean formula is valid if and only if its translation into an $\mathcal{ALC}$ concept description is coherent, we assign *levels* to the variables occurring in constraint trees as follows:

(1) The concept variable that is the root of the constraint tree has the level 0.
(2) If the constraint tree contains a constraint $X(\exists R)Y$ or $X(\forall R)Y$ and X has the level n, then Y has the level $n+1$.
(3) If the constraint tree contains a constraint $X \sqsubseteq Y \sqcup Z$, then X, Y and Z all have the same level.
(4) If the constraint tree contains a constraint x: X, then x and X have the same level.
(5) If the constraint tree contains a constraint xRy and x has the level n, then y has the level $n+1$.

This defines unique level assignments for constraint trees obtained from fresh constraint trees by finitely many applications of the completion rules.

Lemma 6.2. *A quantified boolean formula $P.M$ is valid if and only if its translation $[P.M]$ is a coherent $\mathcal{ALC}$ concept description.*

Proof. Let $P.\{C_1, \ldots, C_n\}$ be a quantified boolean formula such that P is a prefix from 1 to m. Furthermore, let $\bar{P} \cup \bar{C}_1 \cup \cdots \cup \bar{C}_n$ be a fresh constraint tree obtainable by unfolding the translation $[P.\{C_1, \ldots, C_n\}]$, where $\bar{P}$ is a fresh constraint tree obtainable by unfolding $[P]$ and, for $i \in 1\,..\,n$, $\bar{C}_i$ is a fresh constraint tree obtainable by unfolding $[C_i]$.

Let $\hat{P}$ be a trace of $\bar{P}$. Then $\hat{P}$ contains exactly one individual variable x_i for every $i \in 0\,..\,m$. Furthermore, $\hat{P}$ contains the chain $x_0Rx_1, \ldots, x_{m-1}Rx_m$. We say that i is *true* in $\hat{P}$ if $\hat{P}$ contains two constraints x_i: X and $X \sqsubseteq A$, and that i is *false* in $\hat{P}$ if $\hat{P}$ contains two constraints x_i: X and $X \sqsubseteq \neg A$. Every $i \in 1\,..\,m$ is either true or false in $\hat{P}$, but not both. Thus $\hat{P}$ defines a P-assignment α as follows: $\alpha(i)=1$ if i is true in $\hat{P}$ and $\alpha(i)=0$ if i is false in $\hat{P}$.

Since no trace of $\bar{P}$ contains a clash, every completion of $\bar{P}$ is clash-free. Furthermore, the set of the P-assignments defined by the traces contained in a completion of $\bar{P}$ is canonical for P. Vice versa, every set of P-assignments that is canonical for P can be obtained from the same completion of $\bar{P}$. This correspondence between canonical sets of assignments and completions is crucial for our proof.

Let C_i be one of the clauses. Then $\bar{C}_i$ contains no existential edges. The leaves of $\bar{C}_i$ correspond exactly to the literals of C_i. If l is a positive literal in C, then the corresponding leaf X of $\bar{C}_i$ has the level l and $\bar{C}_i$ contains the constraint $X \sqsubseteq A$. If l is a negative literal in C, then the corresponding leaf X of $\bar{C}_i$ has the level $-l$ and $\bar{C}_i$ contains the constraint $X \sqsubseteq \neg A$.

"$[P.M]$ coherent $\Rightarrow$ $P.M$ valid". (1)

Suppose $[P.\{C_1, \ldots, C_n\}]$ is a coherent $\mathcal{ALC}$ concept description. Then $\bar{P} \cup \bar{C}_1 \cup \cdots \cup \bar{C}_n$ has a clash-free completion $\tilde{P} \cup \tilde{C}_1 \cup \cdots \cup \tilde{C}_n$ such that $\tilde{P}$ is a completion of $\bar{P}$.

Let $\hat{P}$ be a trace of $\bar{P}$ such that $\hat{P} \subseteq \tilde{P}$ and let α be the P-assignment defined by $\hat{P}$. Furthermore, let C_i be one of the clauses. It suffices to show that C_i contains a literal that is satisfied by α.

Let $\hat{P} \cup \hat{C}_i$ be the clash-free trace of $\bar{P} \cup \bar{C}_i$ such that $\hat{C}_i \subseteq \tilde{C}_i$. Then $\hat{C}_i$ contains exactly one constraint x: X such that X is a leaf. Since $\hat{C}_i$ is clash-free, α satisfies the literal in C_i that corresponds to X.

$$\text{“}P.M \text{ valid} \Rightarrow [P.M] \text{ coherent”}. \tag{2}$$

Suppose $P.\{C_1, \ldots, C_n\}$ is valid. Then there exists a set A of P-assignments that is canonical for P such that every $\alpha \in \mathrm{A}$ satisfies every clause C_i. Let $\tilde{P}$ be a completion of $\bar{P}$ that yields A. It suffices to show that there exists a clash-free completion $\tilde{P} \cup \tilde{C}_i$ of $\tilde{P} \cup \bar{C}_i$ for every clause C_i since the union of these completions is a clash-free completion of $\tilde{P} \cup \bar{C}_1 \cup \cdots \cup \bar{C}_n$ (because no $\bar{C}_i$ contains an existential edge and every individual variable having a level between 1 and m is either true or false in $\tilde{P}$).

Let C_i be one of the clauses and let $\rightarrow_Q$ be the following restriction of the completion rule $\rightarrow_{\sqcup}$:

if $X \sqsubseteq Y_1 \sqcup Y_2$, x: X, x: Z, $Z \sqsubseteq C$ and $Y_1 \sqsubseteq D$ are in S,

then add x: Y_1 if $C = D$ and x: Y_2 if $C \neq D$.

By applying the completion rules $\rightarrow_\forall$ and $\rightarrow_Q$ to $\tilde{P} \cup \bar{C}_i$ we obtain a completion $\tilde{P} \cup \tilde{C}_i$ of $\bar{P} \cup \bar{C}_i$. It remains to show that $\tilde{P} \cup \tilde{C}_i$ is clash-free.

Suppose $\tilde{P} \cup \tilde{C}_i$ contains a clash. Let $x_0Rx_1, \ldots, x_{k-1}Rx_k$ be constraints in $\tilde{P}$ such that x_0 has the level 0 and x_k is involved in a clash in $\tilde{P} \cup \tilde{C}_i$. Because $\tilde{P} \cup \tilde{C}_i$ was obtained using the completion rule $\rightarrow_Q$, the greatest level in $\tilde{C}_i$ must be k. Let $\hat{P} \subseteq \tilde{P}$ be a trace of $\bar{P}$ containing the constraints $x_0Rx_1, \ldots, x_{k-1}Rx_k$ and let $\alpha \in \mathrm{A}$ be the P-assignment defined by $\hat{P}$. Now it is easy to verify that α satisfies no literal in C_i. This contradicts our assumption that every assignment of A satisfies every clause. □

Theorem 6.3. *Deciding coherence and subsumption of $\mathcal{ALC}$ concept descriptions are PSPACE-complete problems.*

Proof. In Section 2 we have shown that the subsumption and incoherence problems can be reduced to each other in linear time. In Section 5 we have shown that coherence can be decided with linear space. Since QBF is PSPACE-complete and the given reduction to the coherence problem can be done in quadratic time, we have the claim by the preceding lemma, which states the correctness of the reduction. □

7. Feature descriptions

There is a second family of attributive concept descriptions known as feature descriptions or feature terms [18, 30, 33, 34], which has been developed for the syntactic description of natural languages with so-called unification grammars [15, 19, 29, 32]. Closely related is Aït-Kaci's ψ-term calculus [1–3], which is geared towards logic programming and knowledge representation. The distinctive difference to $\mathcal{ALC}$ descriptions is that feature descriptions employ only functional roles, called *features*, which results in very different computational properties. In particular, the so-called agreement construct causes undecidability with roles but doesn't with features. In this section we review feature descriptions and compare them with $\mathcal{ALC}$ descriptions. A survey of both terminological representation systems and unification grammars pointing out their similarities and differences can be found in [26].

We first define the agreement construct for roles. For this it is convenient to take a different, but equivalent, look at roles.

Let M be a set. Then a *set function* on M is a total function

$$f: 2^M \rightarrow 2^M$$

such that

$$f(N) = \bigcup_{a \in N} f(\{a\})$$

for every subset $N \subseteq M$. Note that a set function on M is given by its values on the singletons of M, and that the composition of two set functions on M is again a set function on M. Now every binary relation r on M defines a set function $r_\uparrow$ on M by

$$r_\uparrow(\{a\}) = \{b \mid (a, b) \in r\},$$

and every set function f on M defines a binary relation $f_\downarrow$ on M by

$$(a, b) \in f_\downarrow \Leftrightarrow b \in f(\{a\}).$$

This defines a bijection between the binary relations on M and the set functions on M since $f_{\downarrow\uparrow} = f$ and $r_{\uparrow\downarrow} = r$.

A *path* is a finite, possibly empty sequence of roles. Given an interpretation $\mathcal{I}$, a path $R_1 \ldots R_n$ denotes the composition of the set functions $(R_1^{\mathcal{I}})_\uparrow, \ldots, (R_n^{\mathcal{I}})_\uparrow$, where $(R_n^{\mathcal{I}})_\uparrow$ is applied first. The empty path is interpreted by $\mathcal{I}$ as the identity function of $2^{D^{\mathcal{I}}}$.

An *agreement* is a concept description taking the form $p \downarrow q$, where p and q are paths. Given an interpretation $\mathcal{I}$, an agreement $p \downarrow q$ denotes the set of all elements on which the paths p and q agree:

$$(p \downarrow q)^{\mathcal{I}} = \{d \in D^{\mathcal{I}} \mid p^{\mathcal{I}}(\{d\}) = q^{\mathcal{I}}(\{d\})\}\,.$$

Agreements are called *role value maps* in [7], where they are introduced as part of KL-ONE. Only recently Schmidt-Schauß [31] showed that subsumption of attributive concept descriptions built from agreements, concepts, intersections and universal role quantifications is undecidable.

A role R is called a *feature* in an interpretation $\mathcal{I}$ if $R^{\mathcal{I}}$ is functional, that is,

$$(a, b) \in R^{\mathcal{I}} \wedge (a, c) \in R^{\mathcal{I}} \;\Rightarrow\; b = c$$

for all $a,b,c \in D^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is called a *feature interpretation* if it interprets every role as a feature.

Let $\mathcal{ALCA}$ be the language obtained from $\mathcal{ALC}$ by adding agreements. The already mentioned result of Schmidt-Schauß [31] implies that coherence and subsumption of $\mathcal{ALCA}$ descriptions with respect to all interpretations are undecidable. However, if only feature interpretations are admitted, deciding coherence of $\mathcal{ALCA}$ descriptions is an NP-complete problem [33, 34]. Consequently, deciding subsumption of $\mathcal{ALCA}$ descriptions with respect to feature interpretations is a co-NP-complete problem. The same complexities already hold for $\mathcal{ALC}$ descriptions. Hence adding agreement in the case where only feature interpretations are admitted doesn't change the complexity while it causes a jump from PSPACE-completeness to undecidability if all interpretations are admitted.

Not surprisingly, feature descriptions were developed for applications where feature agreements are essential while applications of terminological representation systems exploit general roles but avoid role agreements. Nevertheless, there are important applications in computational linguistics (for instance, coordination) that could be given better solutions using general roles (in this context usually known as set-valued attributes), and agreements could significantly extend the applicability of terminological representation systems.

It is possible to have a decidable language that offers roles as well as agreements for features. To this purpose one introduces different symbols for (general) roles and features and admits only those interpretations that interpret features with functional relations. Furthermore, agreements are only available for paths containing only features. Hollunder [13] recently showed that such a modification of $\mathcal{ALCA}$ has a decidable subsumption relation.

A crucial technical difference between features and roles is the fact that features allow for a disjunctive normal form for concept descriptions while general roles do not. To see this, note that the equivalences

$$(C \sqcup D) \sqcap E \sim (C \sqcap E) \sqcup (D \sqcap E)$$

$$\exists R\colon (C \sqcup D) \sim (\exists R\colon C) \sqcup (\exists R\colon D)$$

$$\forall R\colon (C \sqcup D) \sim (\forall R\colon C) \sqcup (\forall R\colon D)$$

all hold for features, but that the last equivalence does not hold for roles. A similar difference is that the equivalence

$$\exists R\colon (C \sqcap D) \sim (\exists R\colon C) \sqcap (\exists R\colon D)$$

holds for features but does not hold for roles. Another important difference is that concept descriptions employing only features can be expressed as quantifier-free formulas of predicate logic [33, 34], while this is not possible for attributive concept descriptions employing general roles.

For unification grammar-based parsing of natural language algorithms [9, 11, 16, 17] have been developed that are reported to perform satisfactorily in practice for feature descriptions with unions or disjunctions. It should be interesting to attempt the extension of these techniques to concept descriptions with general roles. One difficulty is that the algorithms for feature descriptions are rather different from the algorithms developed in this paper. In particular, our coherence checking algorithm for $\mathcal{ALC}$ is not incremental, that is, doesn't produce a simplified version of the checked description. The algorithms for feature descriptions are incremental and are thus much better suited for practical applications.

Acknowledgement

We are grateful to Bernhard Nebel for inspiring discussions. The research reported in this paper was done while Schmidt-Schauß was with the DFKI in Kaiserslautern and Smolka was with the IWBS of IBM Deutschland in Stuttgart. Smolka was funded by the EUREKA Project Protos (EU 56).

References

[1] H. Aït-Kaci, An algebraic semantics approach to the effective resolution of type equations, *Theor. Comput. Sci.* **45** (1986) 293–351.

[2] H. Aït-Kaci, A lattice-theoretic approach to computation based on a calculus of partially ordered type structures, Ph.D. Thesis, University of Pennsylvania, Philadelphia, PA (1984).

[3] H. Aït-Kaci and R. Nasr, LOGIN: a logic programming language with built-in inheritance, *J. Logic Program.* **3** (1986) 185–215.

[4] A. Borgida, R.J. Brachman, D.L. McGuiness and L.A. Resnick, CLASSIC: a structural data model for objects, in: *Proceedings ACM SIGMOD International Conference on Management of Data*, Portland, OR (1989) 59–67.

[5] R.J. Brachman and H.J. Levesque, The tractability of subsumption in frame-based description languages, in: *Proceedings AAAI-84*, Austin, TX (1984) 34–37.

[6] R.J. Brachman, V. Pigman Gilbert and H.J. Levesque, An essential hybrid reasoning system: knowledge and symbol level accounts in KRYPTON, in: *Proceedings IJCAI-85*, Los Angeles, CA (1985) 532–539.

[7] R.J. Brachman and J.G. Schmolze, An overview of the KL-ONE knowledge representation system, *Cogn. Sci.* **9** (2) (1985) 171–216.

[8] F. Donini, B. Hollunder, M. Lenzerini, A.M. Spaccamela, D. Nardi and W. Nutt, The complexity of existential quantification in terminological reasoning, DFKI-Rept., DFKI, Kaiserslautern, FRG (to appear).

[9] J. Dörre and A. Eisele, Determining consistency of feature terms with distributed disjunctions, in: D. Metzing, ed., *GWAI-89, 13th German Workshop on Artificial Intelligence*, Informatik Fachberichte **216** (Springer, Berlin, 1989) 270–279.

[10] J. Doyle and R.S. Patil, Language restrictions, taxonomic classifications, and the utility of representation services, Tech. Memo MIT/LCS/TM387, Laboratory for Computer Science, MIT, Cambridge, MA (1989); also *Artif. Intell.* (1991).

[11] A. Eisele and J. Dörre, Unification of disjunctive feature descriptions, in: *Proceedings 26th Annual Meeting of the ACL*, Buffalo, NY (1988) 286–294.

[12] M.R. Garey and D.S. Johnson, *Computers and Intractability—A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, CA, 1979).

[13] B. Hollunder and W. Nutt, Subsumption algorithms for concept description languages, DFKI-Res. Rept. RR-90-04, DFKI, Kaiserslautern, FRG (1990).

[14] T.S. Kaczmarek, R. Bates and G. Robins, Recent developments in NIKL, in: *Proceedings AAAI-86*, Philadelphia, PA (1986) 978–987.

[15] R.M. Kaplan and J. Bresnan, Lexical-functional grammar: a formal system for grammatical representation, in: J. Bresnan, ed., *The Mental Representation of Grammatical Relations* (MIT Press, Cambridge, MA, 1982) 173–381.

[16] R.T. Kasper, Feature structures: a logical theory with applications to language analysis, Ph.D. Thesis, University of Michigan, Ann Arbor, MI (1987).

[17] R.T. Kasper, A unification method for disjunctive feature descriptions, in: *Proceedings 25th Annual Meeting of the ACL*, Stanford, CA (1987) 235–242.

[18] R.T. Kasper and W.C. Rounds, A logical semantics for feature structures, in: *Proceedings 24th Annual Meeting of the ACL*, New York (1986) 257–265.

[19] M. Kay, Functional grammar, in: *Proceedings Fifth Annual Meeting of the Berkeley Linguistics Society*, Berkeley, CA (1979).

[20] H.J. Levesque and R.J. Brachman, Expressiveness and tractability in knowledge representation and reasoning, *Comput. Intell.* **3** (1987) 78–93.

[21] H.J. Levesque and R.J. Brachman, A fundamental tradeoff in knowledge representation and reasoning (revised version), in: R.J. Brachman and H.J. Levesque, eds., *Readings in Knowledge Representation* (Morgan Kaufmann, Los Altos, CA, 1985) 41–70.

[22] R. MacGregor and R. Bates, The Loom knowledge representation language, Tech. Rept. ISI/RS-87-188, University of Southern California, Information Science Institute, Marina del Rey, CA (1987).

[23] B. Nebel, Computational complexity of terminological reasoning in BACK, *Artif. Intell.* **34** (1988) 371–383.

[24] B. Nebel, Reasoning and revision in hybrid representation systems, Ph.D. Thesis, Universität des Saarlandes, Saarbrücken, FRG (1989); also in: Lecture Notes in Artificial Intelligence (Springer, Berlin, to appear).

[25] B. Nebel, Terminological reasoning is inherently intractable, IWBS Rept. 82, IWBS, IBM Deutschland, Stuttgart, FRG (1989); also *Artif. Intell.* **43** (1990) 235–249.

[26] B. Nebel and G. Smolka, Representation and reasoning with attributive descriptions, IWBS Rept. 81, IWBS, IBM Deutschland, Stuttgart, FRG (1989); also in: K.H. Bläsius, U. Hedtstück and C.-R. Rollinger, eds., *Sorts and Types in Artificial Intelligence*, Lecture Notes in Artificial Intelligence **418** (Springer, Berlin, 1990) 112–139.

[27] B. Nebel and K. von Luck, Hybrid reasoning in BACK, in: Z.W. Ras and L. Saitta, eds., *Methodologies for Intelligent Systems* (North-Holland, Amsterdam, 1988) 260–269.

[28] P.F. Patel-Schneider, Small can be beautiful in knowledge representation, in: *Proceedings IEEE Workshop on Principles of Knowledge-Based Systems*, Denver, CO (1984) 11–16.

[29] C. Pollard and I.A. Sag, *An Information-Based Syntax and Semantics*, CSLI Lecture Notes **13** (Center for the Study of Language and Information, Stanford, CA, 1987).

[30] W.C. Rounds and R.T. Kasper, A complete logical calculus for record structures representing linguistic information, in: *Proceedings First IEEE Symposium on Logic in Computer Science*, Boston, MA (1986) 38–43.

[31] M. Schmidt-Schauß, Subsumption in KL-ONE is undecidable, in: R.J. Brachman, H.J. Levesque and R. Reiter, eds., *Proceedings First International Conference on Principles of Knowledge Representation and Reasoning*, Toronto, Ont. (1989) 421–431.

[32] S.M. Shieber, *An Introduction to Unification-Based Approaches to Grammar*, CSLI Lecture Notes **4** (Center for the Study of Language and Information, Stanford, CA, 1986).

[33] G. Smolka, Feature constraint logics for unification grammars, IWBS Rept. 93, IWBS, IBM Deutschland, Stuttgart, FRG (1989); also *J. Logic Program.* (to appear).

[34] G. Smolka, A feature logic with subsorts, LILOG Rept. 33, IWBS, IBM Deutschland, Stuttgart, FRG (1988).

[35] M.B. Vilain, The restricted language architecture of a hybrid representation system, in: *Proceedings IJCAI-85*, Los Angeles, CA (1985) 547–551.