This is a reconstruction of the paper

Subsumption in KL-ONE is Undecidable

from the old MS-Word files, published in 1989

Due to Microsoft's permanently changing MS-Word-standards this must be clumsily done by hand. The paper is now typesetted using LaTeX.
The layout may have changed slightly, the page numbers are different, but there is (almost) no other correction or change, in particular the numbering of sections, lemmas, theorems and corollaries is the same.

The current address of the author is:

Manfred Schmidt-Schauß
Institut für Informatik,
J.-W.-Goethe-Universität,
Postfach 11 19 32,
D-60054 Frankfurt,
Germany

Tel: (+49)69-798-28597,
Fax: (+49)69-798-28919,
E-mail: schauss@ki.informatik.uni-frankfurt.de

Subsumption in KL-ONE is Undecidable

Manfred Schmidt-Schauß
DFKI, Universität Kaiserslautern
PO-box 3049, D-6750 Kaiserslautern
F.R. Germany

Abstract. It is shown that in the frame-based language KL-ONE it is undecidable whether one concept is subsumed by another concept. In fact a rather small sublanguage of KL-ONE called $\mathcal{ALR}$ is used which has only the concept forming operators conjunction, value restriction, and role value maps using only '='. In particular, number restrictions are not used. The language $\mathcal{ALR}$ can be viewed as an extension of feature terms without complements and unions, where features have sets as values instead of elements. Our result shows that there is a basic difference between feature terms and KL-ONE, since the complexity of subsumption switches from quasi-linear to undecidable if the restriction is dropped that roles are functional.

1 Introduction

The knowledge representation language KL-ONE [Brachman and Schmolze, 1985, Brachman et. al., 1985, Schmolze and Israel, 1983] permits describing concepts on the basis of unary predicates (concepts, frames) and binary predicates (roles, slots). It is a language supporting the definition and exploitation of taxonomical knowledge including multiple inheritance, and has as advantage a declarative semantics that permits to compare implementations. Several KL-ONE-related languages are currently being used as the basis of knowledge representation systems [Brachman and Schmolze, 1985, von Luck et. al. 1987, Kaczmarek et. al., 1986, Mac Gregor, 1988], in particular for natural language processing.

The terminological component (T-Box) of KL-ONE gives a potential user the possibility to structure a domain of interest by using concepts and roles (frames and slots). Usually, such a description starts by postulating certain primitive concepts and roles, and afterwards defining concepts using the operators available. An example for such concepts may be person, animal, female, male, thing, and examples for roles are child (has-child) and father (father-of). New concepts can be defined via conjunction ("a person that is female"), via value restriction with respect to some role ("a person with every male friend is a doctor"), via number restriction ("a person with more than three children"), via role value maps ("a man with every child of a child of his father is also a child"), and some other formation possibilities. It is also possible to restrict models by some axioms, for example by requiring that certain concepts are disjoint or that some concepts form a covering of another concept.

Research in computational linguistics has lead to a related knowledge representation method as the basis of unification grammars [Kay, 1985, Shieber, 1986], also called feature structures. H. Aït-Kaci has independently developed a very similar mechanism [Aït-Kaci, 1984, Aït-Kaci and Nasr, 1986] with the intention to apply it in knowledge representation systems. The so-called Ψ-terms are defined and implemented in a Logic Programming language. A similar device are the feature terms, described in [Smolka and Aït-Kaci, 1987, Smolka, 1988], where concepts are called sorts and roles are called features. We give an example for feature terms: Let **car** be a sort symbol (concept) and let `speed` and `age` be features applicable to cars. Then (`car` $\sqcap$ (`age` $\downarrow$ `speed`)) is a feature term denoting the set of all cars that have equal age and speed. A feature term used in linguistic application may be (**sentence** $\sqcap$ ((`subj number`) $\downarrow$ (`pred number`))), which denotes sentences with agreeing number of subject and number of predicate. The difference to knowledge representation á la KL-ONE is that features denote partial functions. The term forming rules for feature terms directly correspond to the concept forming possibilities conjunction, value restriction and role-value maps available in KL-ONE. In addition, feature terms may be defined using a predefined lattice structure on sorts (the primitive concepts). Indeed, the expressiveness of feature terms can be viewed as a subset of KL-ONE where all roles are functional.

The basic service provided by all these languages is a reasoning facility called 'classification' that informs the user whether one of his already defined concepts is subsumed by some other already defined concept. H. Levesque and R. Brachman strongly argue in several papers [Brachman and Levesque, 1984, Levesque and Brachman, 1985, Levesque and Brachman, 1987] that classification should be executable in polynomial time. They showed that there is a tradeoff between expressivity of the concept description language and the complexity of the subsumption test. In particular they proved that reasonably expressive concept description languages have a co-NP-hard subsumption problem.

In the feature term language the main inference is called 'unification'. Basically, unification of two feature terms amounts to computing a simplified representation of their conjunction and a test, whether this conjunction denotes a nonempty set. The test for consistency is in fact a classification problem. It was shown by H. Aït-Kaci [1984], that classification can be performed in quasi-linear time. Recently it was shown that the classification problem for feature terms including negation is co-NP-complete [Smolka, 1988].

In my opinion, knowledge representation formalisms that don't admit a polynomial classification algorithm are nevertheless useful, since for example most NP-complete problems permit algorithms that require polynomial time in average. However, a formalism with an undecidable subsumption is unsatisfactory, since in this case it may even be the case that consistency of concepts is not recursively enumerable. Using a formalism with undecidable subsumption in practice means that the corresponding classification is either incomplete or that further constraints to restrict concept formation are to be employed. If the latter is the case, then these restriction should show up in the theory, possibly yielding a decidable subsumption for the restricted language. A further argument for decidability of subsumption is that only in this case it is sensible to combine such a formalism with a first-order language (A-Box) with the intention to exploit the mechanisms and algorithms of the knowledge representation formalism, since otherwise the T-Box is already as powerful as any programming language. However, there are some recent approaches using feature terms as constraints [Höhfeld and Smolka, 1988, Bläsius and Hedtstück, 1988]. These methods use a lazy classification and can thus tolerate undecidable subproblems by postponing the decision until further information is available. Both approaches are restricted to feature terms, however, an extension to KL-ONE-like concept terms appears possible.

Although there is agreement that role value maps provide complications, there are systems allowing them as descriptive possibility and use them also in the (incomplete) classification algorithm. For example NIKL [Kaczmarek et. al., 1986] admits the full expressive power of role value maps, whereas for example in BACK [von Luck et. al., 1987] role value maps are strongly restricted. There are several knowledge representation systems using feature terms and role value maps in the restricted form of agreements. Since for feature terms it is known that the consistency check is quasilinear, and co-NP-complete if complements are permitted, the demand which suggests itself is to extend features to arbitrary binary predicates (or roles).

In this paper it is shown that subsumption in $\mathcal{ALR}$, a rather small sublanguage of KL-ONE is undecidable. This result was a surprise, since the very similar language of feature terms has a tractable classification problem, and my and other peoples expectation was that extending feature terms by set-valued features would only moderately increase the complexity of classification. This shows that role value maps in the restricted form are acceptable, but that their general form as used in KL-ONE should be restricted.

For this result, conjunction, value restriction, and role value maps using only $=$ are needed, but not more. The proof uses the undecidability of the word problem in groups. This is more convenient than the word problem in semigroups, since using the latter one can only show undecidability of subsumption if $\subseteq$ is permitted as comparison operator in role value maps.

Recently, P. F. Patel-Schneider [1989] has shown that classification in NIKL is undecidable. His sublanguage of KL-ONE requires more of the expressivity of KL-ONE than our sublanguage, such as role value maps with '$\subseteq$' as comparison operator, inverse roles and number restrictions. In the paper it is mentioned that inverse roles can be omitted, but role value maps with '$\subseteq$' seem to be indispensible. K. Schild [Schild, 1988] presents a proof showing that in a KL-ONE related language that supports only role-definition subsumption of roles is undecidable. The language permits the definition of complements of roles, composition of roles and conjunction of roles.

The paper is structured as follows: In section 2 the syntax and semantics of $\mathcal{ALR}$ and of $\mathcal{ALRC}$ are given. In section 3 we give the proof of undecidability of subsumption by reducing the word problem in groups to it and furthermore discuss some consequences.

2 The Language $\mathcal{ALR}$

In the following we describe the syntax and semantics of the sublanguage $\mathcal{ALR}$ of the terminological language KL-ONE [Brachman and Schmolze, 1985, Schmolze and Israel, 1983] as well as of an extension $\mathcal{ALRC}$ of $\mathcal{ALR}$ in a slightly modified linear syntax, but the same semantics. Our language $\mathcal{ALR}$ allows to construct concept expressions by the constructors given below, but it does not support the definition of roles. Consequently we are mainly interested in concepts and their relations.

There are disjoint sets of role symbols and concept symbols. Concept expressions are:

i) concept symbols

ii) $C \sqcap D$, if C and D are concept expressions.

iii) $\forall R : C$, if R is a role symbol and C is a concept expression.

iv) $P = Q$ if P and Q are lists of roles.

Concept expressions in $\mathcal{ALR}$ are composed from symbols and the three methods of construction, possibly by inserting some brackets when necessary. The concept expression for the concept "a man with every child of a child of his father is also a child" would be something like `(man` $\sqcap$ `((father, child, child) = (child)))`. In the NIKL-syntax this would be `(and man (all (compose father child child) child) (all child (compose father child child) child))`.

In the proofs we will use the more expressive language $\mathcal{ALRC}$ that extends the language $\mathcal{ALR}$ by the following three formation rules for concepts:

v) $C \sqcup D$, if C and D are concept expressions.

vi) $\exists R : C$, if R is a role symbol and C is a concept expression.

vii) $\neg C$ if C is a concept expression.

The language $\mathcal{ALRC}$ is an extension of the attributive concept description language $\mathcal{ALC}$ [Schmidt-Schauß and Smolka, 1988] by role value maps.

We need some facts about relations, compositions of relations and applications of relations to sets and elements. Let R, S be relations over a set M, i.e., $R, S \subseteq M \times M$. The composition of R and S is defined as:

$$R \circ S := \{(x, y) | \exists z \in M : (x, z) \in R \wedge (z, y) \in S\}.$$

The application of a relation R to a set $s \subseteq M$ is defined as follows:

$$sR := \{y | \exists x : x \in s \wedge (x, y) \in R\}.$$

The application of R to an element x is defined analogously:

$$xR := \{y | (x, y) \in R\}.$$

Obviously, we have $s(R \circ S) = (sR)S$ and $x(R \circ S) = (xR)S$ for a set s and an element x.

The semantics of the roles, concept symbols and concept expressions is as usual [Levesque and Brachman, 1985, Levesque and Brachman, 1987, von Luck et. al., 1987]. We give the semantics of the more general language $\mathcal{ALRC}$, which includes the semantics for $\mathcal{ALR}$.

An *interpretation* I is a pair (M, I), where M is a set and I an interpretation function, such that

i) for every concept symbol $C : I(C) \subseteq M$.
ii) for every role symbol $R : I(R) \subseteq M \times M$.

Lists of roles are interpreted as the composition of relations:

$$I((R_1, \ldots, R_n)) = I(R_1) \circ \ldots \circ I(R_n).$$

We interpret defined concepts as subsets of M as follows:

$$\begin{array}{ll} I(C \sqcap D) & = I(C) \cap I(D) \\ I(C \sqcup D) & = I(C) \cup I(D) \\ I(\forall R : C) & = \{x \in M | \forall y : (x, y) \in I(R) \Rightarrow y \in I(C)\} \\ I(\exists R : C) & = \{x \in M | \exists y : (x, y) \in I(R) \wedge y \in I(C)\} \\ I(P = Q) & = \{x \in M | x(I(P)) = x(I(Q))\} \\ I(\neg C) & = M \setminus I(C) \end{array}$$

where C and D are concepts, R is a role symbol, and P and Q are lists of roles.

Subsumption and Consistency are defined with respect to this semantics:

- A concept expression C **subsumes** a concept expression D, iff for all interpretations I, we have $I(C) \supseteq I(D)$.
- A concept C is **consistent**, iff there exists an interpretation I, such that $I(C) \neq \emptyset$, otherwise C is called **inconsistent**.
- Two concepts C and D are **equivalent**, iff C subsumes D and D subsumes C.

Since we have negation in $\mathcal{ALRC}$, and since the empty concept can be defined, subsumption problems in $\mathcal{ALRC}$ are equivalent to inconsistency problems.

2.1 Lemma. Let C, D be concept expressions with respect to $\mathcal{ALRC}$. Then the following three statements are equivalent:

i) D subsumes C,
ii) $\neg D \sqcap C$ is an inconsistent concept.
iii) $\neg D \sqcap C$ is subsumed by $\neg D \sqcap D$ (the empty concept).

Proof. The concept D subsumes C, iff $I(D) \supseteq I(C)$. This in turn is equivalent to $(M \setminus I(D)) \cap I(C) = \emptyset$. Thus i) is equivalent to ii). Obviously, $I(\neg D \sqcap D) = \emptyset$ for all interpretations I, hence ii) is equivalent to iii). $\Box$

Subsumption is not equivalent to consistency of concepts in the language $\mathcal{ALR}$, since in $\mathcal{ALR}$ all concepts are consistent. Even if $\exists R : C$ and the role-defining operator (*restrict R C*) from NIKL are permitted, then all concepts remain consistent, which follows by considering a one-element interpretation

$I = (M, I)$ with $M = \{m\}$, where all concept symbols are interpreted as M and all roles as $\{(m,m)\}$. In this interpretation, the restriction of a role has no effect, and all defined concepts have M as extension. Thus all concepts remain consistent.

Let us compare the expressiveness of $\mathcal{ALR}$ and $\mathcal{ALRC}$ with that of other knowledge representation languages:

Obviously, $\mathcal{ALR}$ is a sublanguage of KL-ONE. $\mathcal{ALR}$ has less expressivity than the sublanguage of NIKL used by P.F.Patel-Schneider [Patel-Schneider, 1989] to show undecidability of subsumption in NIKL. The $\mathcal{ALR}$-concept $\forall R : C$ is equivalent to `(all R (restrict R C))` in NIKL and to `(all R C)` in $\mathcal{FL}$ [Levesque and Brachman, 1985]. A concept description using role value maps $((R_1, \ldots, R_n) = (S_1, \ldots, S_m))$ is equivalent to the NIKL-expression `(and (all (compose` $R_1 \ldots, R_n$`) (compose` $S1, \ldots, S_m$`) (all (compose` $S_1, \ldots, S_m$`) (compose` $R_1, \ldots, R_n$`)))`. I conjecture that the expressivity of $\mathcal{ALR}$ is strictly smaller, since it is not possible to define functional roles in $\mathcal{ALR}$.

There is a close relationship between $\mathcal{ALR}$ and feature terms. $\mathcal{ALR}$ can be seen as feature terms (without negation and union) with the semantics of features changed from (partial) functional to arbitrary binary relations. The same relationship holds between $\mathcal{ALRC}$ and feature terms with complements and union. Of course, the descriptive power is not really comparable, since for example the rules for computing with feature terms and concept expressions in $\mathcal{ALRC}$ are different [Smolka, 1988, Schmidt-Schauß and Smolka, 1988]. For example the rule $\forall R : (C \sqcup D) \rightarrow (\forall R : C) \sqcup (\forall R : D)$ is valid for feature terms, whereas this is false in $\mathcal{ALRC}$.

3 Subsumption in $\mathcal{ALR}$ is Undecidable

In this section we will show that subsumption in $\mathcal{ALR}$ is undecidable by reducing the word-problem in groups to it. Since the semantics of $\mathcal{ALRC}$ and $\mathcal{ALR}$ are compatible, it is sufficient to show that subsumption of $\mathcal{ALR}$-expressible concept expressions is undecidable in $\mathcal{ALRC}$. Hence we will use the language $\mathcal{ALRC}$ in the following, but we will encode the subsumption problem using operators from $\mathcal{ALR}$.

Similar as in [Schmidt-Schauß and Smolka, 1988, Smolka, 1988], we transform subsumption problems of $\mathcal{ALR}$-concepts into a system C of constraints, where every single constraint is of one of the forms $s \subseteq C, x \in C, x \in s, s = t$. We write x, y, z for element variables, s, t for expressions of the form $xR_1 \ldots R_n$, and C, D for concept expressions. The reason for using constraints is that the proofs are far more readable than in linear syntax manipulating concept expressions and that subsumption algorithms can be described in an elegant way (cf. [Schmidt-Schauß and Smolka, 1988]).

Let $I = (M, I)$ be an interpretation. Let α be an assignment of elements in M to element variables of C. We assume that α extends the interpretation function I, i.e., $\alpha C = I(C), \alpha(xP) := (\alpha x)I(P)$ for a concept C and a list of roles P.

Then we say α **satisfies** C, if the following holds:
for $(x \in A) \in C$, we have $\alpha x \in \alpha A$
for $(A \subseteq B) \in C$, we have $\alpha A \subseteq \alpha B$
for $(A = B) \in C$, we have $\alpha A = \alpha B$.

A constraint system C is **consistent**, iff there exists an interpretation I and an assignment α with respect to this interpretation such that α satisfies C. Otherwise C is called **inconsistent**.

3.1 Lemma. Let C be a concept expression. Then C is consistent, iff the constraint system $\{x \in C\}$ is consistent. $\square$

There are several rules for replacing concepts and constraints which make life easier, and which preserve consistency and inconsistency of constraint systems (cf. [Schmidt-Schauß and Smolka, 1988, Smolka, 1988]):

$$\begin{array}{ll}
\neg(\forall R : C) & \leftrightarrow \exists R : \neg C \\
\neg(\exists R : C) & \leftrightarrow \forall R : \neg C \\
x \in A \sqcap B & \leftrightarrow x \in A, x \in B \\
x \in \forall R : C & \leftrightarrow xR \subseteq C \\
x \in \exists R : C & \leftrightarrow y \in xR, y \in C \\
 & \quad \text{where } y \text{ is a new variable} \\
x \in (P = Q) & \leftrightarrow xP = xQ
\end{array}$$

In order to show undecidability of subsumption, we use the undecidability of the word problem in groups [Boone, 1959, Novikov, 1955, Stillwell, 1982]. Such a problem looks as follows: Let $R_1, \ldots, R_{n'}$ be the symbols of a group and let $P_1 = Q_1, \ldots, P_{m'} = Q_{m'}$ be the generating relations of some group. Then the word problem is to test given strings P and Q, whether $P = Q$ is derivable from these relations and the axioms for a group. Our aim is to show that there is a subsumption problem that is equivalent to this word problem.

In order to avoid clumsy notation for the used groups, we assume that only associativity is built-in and that the semi-group-defining relations imply that the semigroup defined by the relations is a group. If we have given generating relations under the assumption that all axioms for a group are built-in, the following procedure gives relations for semigroups ensuring that the generated semigroup is an isomorphic group: Add a new symbol R_e standing for the unit, and add for every symbol R_i a new symbol R_i^- for the inverse. Then add the relations: $R_e \circ R = R$ and $R \circ R_e = R$ for every symbol $R \in \{R_e, R_i, R_i^-, i = 1, \ldots, n'\}$ and add the relations $R_i^- \circ R_i = R_e$ and $R_i \circ R_i^- = R_e$ for all symbols R_i. The defining relations for the group are translated as follows: We assume that the words occurring in the relations are composed of symbols or inverses of symbols. The unit is translated into R_e, symbols are translated identically and inverses of symbols are translated by $(R_i)^{-1} \to R_i^-$. Now the new relations together with the translated generated relations defining the group provide a semigroup, which is isomorphic to the original group and has an equivalent word problem.

These considerations permit us to assume in the following that the symbols are $R_1, \ldots, R_n$ and that the relations are $P_1 = Q_1, \ldots, P_m = Q_m$, where the group defining relations are among these relations. By $\mathcal{G}$ we denote the free semigroup (which is in fact a group) generated by the symbols $R_1, \ldots, R_n$ and the relations $P_1 = Q_1, \ldots, P_m = Q_m$. This is the semigroup consisting of all congruence-classes of words from $\{R_1, \ldots, R_n\}^*$, where two words are congruent, if the least congruence on $\{R_1, \ldots, R_n\}^*$ that contains the relations makes them congruent. For convenience we denote elements of $\mathcal{G}$ by $[P]$, where P is a string of symbols and $[P]$ denotes the congruence class with respect to the defining relations.

We say an equation $P = Q$ is derivable from the relations, denoted as $\vdash P = Q$, if it can be generated from the relations using the following rules:

1) $\vdash P_i = Q_i$ for all generating relations.
2) $\vdash P = P$ for all words P from $\{R_1, \ldots, R_n\}^*$.
3) $\vdash P = Q$ if $\vdash Q = P$
4) $\vdash P = Q$ if $\vdash P = P'$ and $\vdash P' = Q$
5) $\vdash PP' = QQ'$ if $\vdash P = Q$ and $\vdash P' = Q'$.

We have $[P] = [Q]$ iff the equation $P = Q$ can be derived from the relations using rules 1) - 5) [Burris and Sankappanavar, 1981, Grätzer, 1979]. Note that this means that for two given words P and Q it is semidecidable, whether they are congruent by using the calculus above for enumeration of all derivable equations. As an example consider the integers, which form a group with respect to addition. We use the usual notation of $0, 1, -1$ and $+$ instead of R_e, R_1, R_1^- and "$\circ$". If all axioms of a group are built-in, then "1" is sufficient as symbol and there are no relations. Considered as semi-group, the group of integers is generated by the three symbols $0, 1, -1$ and the relations $0 + 1 = 1$, $1 + 0 = 1$, $1 + 1 = 1 + 1$, $0 + (-1) = -1$, $(-1) + 0 = -1$, $(-1) + (-1) = (-1) + (-1)$, $1 + (-1) = 0$, $(-1) + 1 = 0$. A derivable equation is for example $1 + (-1) = (-1) + 1$.

In the following we sometimes use the symbols $R_i, i = 1, \ldots, n$ from above also as role symbols. We need an additional role symbol R that is not among these role symbols. Furthermore we use P_i, Q_i, P, and Q also for the lists of roles. Thus it depends on the context whether P or Q is meant as a word in the group or as a list of role symbols.

Now we define several concepts that are needed for the subsumption problem encoding the word problem. For convenience we use $\sqcap$ as associative operator and write lists of roles as composition.

$$C_1 := (R \circ R_1 = R) \sqcap \ldots \sqcap (R \circ R_n = R)$$
$$C_2 := \forall R : (P_1 = Q_1) \sqcap \ldots \sqcap \forall R : (P_m = Q_m)$$

Now let $C := C_1 \sqcap C_2$ and let $D_{P,Q} := \forall R : (P = Q)$.
The subsumption problem, which we are interested in is whether $D_{P,Q}$ subsumes C.

The idea of the construction is to view the relations in the role value maps as relations that define a semigroup and then to make deductions using the deduction rules 1) - 5) above. The concept C_2 encodes the relations that are used to define a particular group, whereas C_1 is only technical; it encodes some fixed point properties that will guarantee the correctness of deducing new role value maps from the given ones similar to deducing new equations from given equations. Let $C_{\mathcal{G}}$ be the following constraint (coming from C):

$$\begin{aligned} C_{\mathcal{G}} := \{ & xR \circ R_1 = xR, \ldots, xR \circ R_n = xR, \\ & xR \subseteq (P_1 = Q_1), \ldots, xR \subseteq (P_m = Q_m)\}, \end{aligned}$$

and let $C_{\mathcal{G}}(P, Q)$ be the following constraint system (coming from $\neg D_{P,Q} \sqcap C$):
$C_{\mathcal{G}}(P, Q) := \{y \in xR, y \in \neg(P = Q)\} \cup C_{\mathcal{G}}$.
Now we can prove the main result as a sequence of lemmas.

3.2 Lemma. $D_{P,Q}$ subsumes C, iff the constraint system $C_{\mathcal{G}}(P, Q)$ is inconsistent:
Proof. Lemma 2.1 yields that $D_{P,Q}$ subsumes C iff $(\neg D_{P,Q}) \sqcap C$ is an inconsistent concept, and Lemma 3.1 yields that this is equivalent to the inconsistency of the constraint system $\{x \in (\neg D_{P,Q}) \sqcap C\}$. Due to the rules above, we can transform this constraint system in several steps as follows:

$$\begin{aligned} & \{x \in \neg(\forall R : (P = Q)), x \in C_1 \sqcap C_2\} \\ \Leftrightarrow & \{x \in \exists R : \neg(P = Q), x \in C_1, x \in C_2\} \\ \Leftrightarrow & \{y \in xR, y_{\neg}(P = Q), x \in C_1, x \in C_2\} \end{aligned}$$

If we develop the concepts C_1 and C_2, then we obtain the constraint system:

$$\begin{aligned} \{ \ & y \in xR, y \in \neg(P = Q), xR \circ R_1 = xR, \ldots, \\ & xR \circ R_n = xR, xR \subseteq (P_1 = Q_1), \ldots, \\ & xR \subseteq (P_m = Q_m)\}, \end{aligned}$$

which is exactly the above defined system $C_{\mathcal{G}}(P, Q)$. Since all used transformations preserve consistency and inconsistency, the lemma holds. □

Now we can prove the crucial fact that the deduction rules above can be simulated in the constraint system $C_{\mathcal{G}}$. The basic idea is the use of the additional constraints $xR \circ R_i = xR$, which permit to view lists of roles in the constraints as words in groups. Without this "technical" addition, this is impossible.

3.3 Lemma. If $P' = Q'$ is derivable from the relations defining $\mathcal{G}$, then for every constraint system C_0 with $C_{\mathcal{G}} \subseteq C_0$, there exists a constraint system $C'_{\mathcal{G}}$ with the following properties:

i) $C_0 \subseteq C'_{\mathcal{G}}$
ii) $\{xR \subseteq (P' = Q')\} \subseteq C'_{\mathcal{G}}$
iii) C_0 is consistent iff $C'_{\mathcal{G}}$ is consistent.

Proof. For a constraint system C let $EQ(C)$ denote the set
$\{P = Q | (xR \subseteq (P = Q)) \subseteq C\}$.
We prove by induction on the length of derivations using the rules 1) - 5) above that given a constraint system C with $C_0 \subseteq C$ and an equation $P' = Q'$ derivable

from $EQ(C)$, the constraint system $\{xR \subseteq (P' = Q')\} \cup C$ is consistent, iff C is consistent. That the consistency of $\{xR \subseteq (P' = Q')\} \cup C$ implies the consistency of C, is obvious and hence not mentioned in the following. Let C be a consistent constraint system containing C_0.

1) If $P' = Q'$ is derivable from $EQ(C)$ with rule 1), then $xR \subseteq (P' = Q')$ is a constraint in C_0, since C_0 contains $C_{\mathcal{G}}$.
2) For every word P from $\{R_1, \ldots, R_n\}^*$, the constraint system $C \cup \{xR \subseteq (P = P)\}$ is consistent, since every assignment α satisfies $xR \subseteq (P = P)$.
3) For an equation $P = Q$ in $EQ(C)$, it is obvious that $C \cup \{xR \subseteq (Q = P)\}$ is consistent, since for every assignment α, we have $\alpha(P = Q) = \alpha(Q = P)$.
4) Let $P = P'$ and $P' = Q$ be in $EQ(C)$ and let α be an assignment that satisfies C. This means $\alpha(xR) \subseteq \alpha(P = P')$ and $\alpha(xR) \subseteq \alpha(P' = Q)$. By the semantics we have that for every element $a \in \alpha(xR) : a(\alpha P) = a(\alpha P')$ and $a(\alpha P') = a(\alpha Q)$. Hence we have also $a(\alpha P) = a(\alpha Q)$, hence $\alpha(xR) \subseteq \alpha(P = Q)$ holds. Thus α satisfies the constraint system $C \cup \{xR \subseteq (Q = P)\}$, hence it is consistent.
5) Let $P = Q$ and $P' = Q'$ be in $EQ(C)$ and let α be an assignment that satisfies C. This means $\alpha(xR) \subseteq \alpha(P = Q)$ and $\alpha(xR) \subseteq \alpha(P' = Q')$. Now the constraints from C_1 have to be used! Since $\alpha(xR \circ R_i) = \alpha(xR)$ for all $i = 1, \ldots, n$, we have also $\alpha(xR \circ P) = \alpha(xR) = \alpha(xR \circ Q)$ by repeated application. For every $a \in \alpha(xR)$ the equation $a(\alpha P) = a(\alpha Q)$ holds. Furthermore $a(\alpha P) \subseteq \alpha(xR)$. Hence for every element $b \in a(\alpha P)$, we have $b(\alpha P') = b(\alpha Q')$. If we take the union of all sets $b(\alpha P')$ and $b(\alpha Q')$, where b ranges over the whole set $a(\alpha P)$, we obtain that $a(\alpha(P \circ P')) = a(\alpha(Q \circ Q'))$ for every $a \in \alpha(xR)$. Hence $\alpha(xR) \subseteq \alpha(P \circ P' = Q \circ Q')$ holds. Thus α is an assignment that satisfies the constraint system $C \cup \{xR \subseteq (P \circ P' = Q \circ Q')\}$, hence this constraint system is consistent. $\square$

3.4 Lemma. Let P and Q be words over $\{R_1, \ldots, R_n\}^*$. Then $[P] = [Q]$ iff $C_{\mathcal{G}}(P, Q)$ is inconsistent.
Proof.
"$\Rightarrow$": Assume by contradiction that $C_{\mathcal{G}}(P, Q)$ is consistent. If $[P] = [Q]$ holds in $\mathcal{G}$, then the rules 1) - 5) are sufficient to derive $P = Q$ from the relations. Lemma 3.3 shows that there exists a consistent constraint system $C'_{\mathcal{G}}$ containing $C_{\mathcal{G}}(P, Q) \cup \{xR \subseteq (P = Q)\}$. The system $C_{\mathcal{G}}(P, Q)$ contains the constraints $y \in xR, y \in \neg(P = Q)$, which contradicts the constraint $\{xR \subseteq (P = Q)\}$. Hence $C_{\mathcal{G}}$ is inconsistent.
"$\Leftarrow$": We show that if $[P] \neq [Q]$, then $C_{\mathcal{G}}(P, Q)$ is consistent. Therefore we assume that $[P] \neq [Q]$. An interpretation $I = (I, M)$ that satisfies $C_{\mathcal{G}}(P, Q)$ can be constructed as follows: Let the domain M be $M := \{a\} \cup \mathcal{G}$, where $a \notin \mathcal{G}$, let $I(R) := \{(a, g) | g \in \mathcal{G}\}$, and let $I(R_i) := \{(g, g \circ [R_i]) | g \in \mathcal{G}\}$ for $i = 1, \ldots, n$. The assignment α is defined such that $\alpha x := a$. This means that $\alpha(xR) = \mathcal{G}$. Since multiplication from right in a group is a bijection, the constraints $xR \circ R_1 = xR, \ldots, xR \circ R_n = xR$ are satisfied. The constraints $xR \subseteq (P_1 = Q_1), \ldots, xR \subseteq (P_m = Q_m)$ are also satisfied, since the equations $[P_i] = [Q_i], i = 1, \ldots, m$ hold

in $\mathcal{G}$. It remains to be shown that $y \in \neg(P = Q)$ and $y \in xR$ can be satisfied. The assignment of the unit $1_\mathcal{G}$ in $\mathcal{G}$ to y, i.e., $\alpha y := 1_\mathcal{G}$ gives an element that is contained in the sets $\alpha(xR)$ and $\alpha(\neg(P = Q)$, since $[P] \neq [Q]$ in $\mathcal{G}$. Hence the thus defined assignment α satisfies the constraint $C_\mathcal{G}(P, Q)$. □

3.5 Theorem. Subsumption in $\mathcal{ALR}$, and hence in KL-ONE, is undecidable.
Proof. Obviously the subsumption problem in Lemma 3.2 can be formulated in $\mathcal{ALR}$. Lemmas 3.2, 3.3, 3.4, show that the concept $D_{P,Q}$ subsumes C, iff $[P] = [Q]$ with respect to the group defined by the relations. Now the well-known result that the word problem in groups is undecidable [Boone, 1959, Novikov, 1955, Stillwell 1982] implies that subsumption is undecidable. □

Note that the reason for using the undecidability of the word problem in groups rather than in semigroups or monoids is that in the proof of Lemma 3.4, the constraints $xR \circ R_i = xR$ have to be satisfied, which requires that multiplication from right must be surjective.

The result in [Boone, 1959, Novikov, 1955, Stillwell, 1982] shows that there exists a group, such that the word problem in this group is undecidable. In our context this means, that there exists a fixed concept C such that it is undecidable whether a given concept $D_{P,Q}$ subsumes C.

3.6 Corollary. In $\mathcal{ALR}$ there exists a fixed concept C, such that it is undecidable, whether a given concept D subsumes C.

As a further corollary of Theorem 3.5 we obtain also that some problems cannot be recursively enumerable, such as non-subsumption in $\mathcal{ALR}$ and consistency of concepts in the language $\mathcal{ALRC}$.

3.7 Corollary. Consistency of concepts in $\mathcal{ALRC}$ is not recursively enumerable.
Proof. Assume for contradiction that the consistent concepts of $\mathcal{ALRC}$ can be recursively enumerated. Since the calculus consisting of the 5 rules for the equations in groups is complete, it follows from Lemma 3.4 that the inconsistent constraints $C_\mathcal{G}(P, Q)$ can be recursively enumerated. Using Lemma 3.1 the assumption that consistent concepts of $\mathcal{ALRC}$ are recursively enumerable implies that the consistent constraint systems $C_\mathcal{G}(P, Q)$ can be recursively enumerated, since $C_\mathcal{G}(P, Q)$ is equivalent to $\{x \in \neg D_{P,Q} \sqcap C\}$. This means inconsistency of $C_\mathcal{G}(P, Q)$ is decidable, which contradicts Lemma 3.4 and Theorem 3.5. □

Of course, this does not hold for $\mathcal{ALR}$ alone, since all $\mathcal{ALR}$-concepts are consistent.

Corollary 3.7 has as a curious consequence that we can give a nonconstructive proof that there must be a consistent concept that denotes the empty set in all finite models:

3.8 Corollary. There exists a consistent $\mathcal{ALRC}$-concept C, such that C denotes a nonempty set only in infinite interpretations.
Proof. Assume, that the corollary is false. Then for every consistent $\mathcal{ALRC}$-concept C, there exists a finite interpretation, such that C is interpreted as a

nonempty set. This implies that consistency of concepts would be recursively enumerable, which contradicts Corollary 3.7. □

In the following we describe some consequences for languages that extend the feature term languages in some way. Note that our proof of undecidability of subsumption in $\mathcal{ALR}$ does not work for the simple feature term language, since R must be interpreted as a proper role, which is not possible in the feature term language.

Let us define the language $\mathcal{ALR}^*$ as an extension of the feature term language. The language $\mathcal{ALR}^*$ has (disjoint) sets of role and feature symbols. The interpretation of roles is as usual, whereas the interpretation of a feature F should be a partial function, i.e.,
$(a, b) \in I(F) \wedge (a, c) \in I(F) \Rightarrow b = c$.
Concept expressions in $\mathcal{ALR}^*$ are:

i) concept symbols
ii) $C \sqcap D$, if C and D are concept expressions.
iii) $\forall R : C$, if R is a role symbol and C is a concept expression.
iv) $F : C$, if F is a feature symbol and C is a concept expression.
v) $P = Q$ if P and Q are lists of roles or features, where the first element may be a role, whereas all other elements in the list are features.

The semantics of concept expressions is slightly changed, the conjunction is (as usual) interpreted as intersection, but the constructs $F : C$ and $P = Q$ include definedness of roles and features. Let $I = (M, I)$ be an interpretation, then

$$\begin{aligned} I(F : C) &:= \{a \in M \mid \exists b \in I(C) : (a, b) \in I(F)\} \\ I(P = Q) &:= \{a \in M \mid (\exists b : b \in I(C) : (a, b) \in I(P)) \wedge aI(P) = aI(Q)\} \end{aligned}$$

The language $\mathcal{ALR}^*$ is a slight extension of the (simple) feature term language. Nevertheless, subsumption is undecidable:

3.9 Theorem. Subsumption in $\mathcal{ALR}^*$ is undecidable.
Proof. The proofs of the lemmas 3.1 and 3.2 remain valid. In the proof of Lemma 3.3 one has to take into account that the semantics has slightly changed. For 1) - 4) there are no problems. In the proof of 5) there are some additions: One has to prove that the assignment α has the additional property that $\alpha(P \circ P' = Q \circ Q') \neq \emptyset$, which holds, since the $\alpha(xR)$ is not empty. The proof of Lemma 3.4 holds also for the modified semantics without changes, since the interpretations for the roles R_i are partial functions in the model construction part. Finally Theorem 3.5 with the modified Lemmas 3.1 – 3.4) can be applied and yields that subsumption in $\mathcal{ALR}^*$ is undecidable. □

An extension of the feature term language which should be investigated is the simple feature term language, where roles are added, but not permitted in role value maps[1], i.e., i) - iv) are as above, but v) is slightly changed:

v') $P = Q$ if P and Q are lists of features.

I conjecture that subsumption in this language remains decidable.

[1] added during lexical correction

4 Conclusion

We have shown that subsumption of concepts in $\mathcal{ALR}$, a considerable small sublanguage of KL-ONE, is undecidable, if the usual standard semantics is used. The reason for this undecidability result seems to be the expressive power of role value maps. They are rather intuitive at first glance, for example they allow the definition of grand-father in terms of the roles father and mother, but provide the full power of a programming language if used excessively.

As mentioned above, subsumption of feature terms is quasi-linear or co-NP-complete, depending on the expressiveness [Aït-Kaci, 1984, Aït-Kaci and Nasr, 1986, Smolka, 1988, Smolka and Aït-Kaci, 1987]. It is remarkable that role value maps in the feature term language do not have such a dramatic effect as in $\mathcal{ALR}$. The main difference between $\mathcal{ALR}$ and the feature term language is that roles in the latter always are functional.

There are several methods to either overcome the problem of undecidability or deal with undecidability.

A first ad-hoc possibility to re-establish decidability is to restrict the expressive power of $\mathcal{ALR}$ or $\mathcal{ALRC}$ by discarding role value maps. A language called $\mathcal{ALC}$, that allows complements in addition but no role value maps has been investigated in [Schmidt-Schauß and Smolka, 1988], where it is shown that subsumption becomes PSPACE-complete in this case. A further possibility is to syntactically restrict role value maps. For example in BACK [von Luck et. al, 1987], the lists of roles in role value maps are restricted to be lists of one element. Another possibility is to permit only role value maps in a role-defining style, i.e., only the form $(R) = (R_1, \ldots, R_n)$ is admissible, and there are no double definitions and no cycles. I suspect that subsumption in $\mathcal{ALR}$ becomes decidable under these restrictions.

Another direction of research is to use another semantics for the used constructs. This is a change of the meaning of the language and hence care should be taken. Nevertheless it may very well be the case that for small concepts the standard semantics fits our intuition, whereas for complex concepts, there may be a choic. Of course, the undecidability result depends on the imposed semantics of complex concepts, i.e., on the asymptotic behaviour of the semantics for large concepts.

Decidability of subsumption of feature terms can be interpreted as a change in the semantics of $\mathcal{ALR}$ or $\mathcal{ALRC}$, respectively. The approach to change the usual first-order logic to a four-valued as in [Patel-Schneider, 1987a, Patel-Schneider, 1987b] may help in re-establishing decidability to the price that the meaning of subsumption has changed. Corollary 3.7 suggests to prevent the formation of concepts that are consistent only in infinite models or to change the semantics such that only finite models are to be considered instead of all including infinite models. This idea is a remedy to Corollary 3.6 since consistency of concepts becomes recursively enumerable with respect to this semantics, but now the status of inconsistency of concepts or constraints system is unclear and can be even worse than before.

A practical useful method is to put subsumption-problems in constraints and use constraint-propagation. The view taken here is that a user is not really interested in the subsumption relation of some complex concepts or in the consistency of concept, which he(she) has typed in, rather in the answer to high-level queries that have as subproblems such subsumption or consistency tests. The idea is that the system computes as much as possible or as much as the user wants and then gives as answer a system C of constraints. These answer can be interpreted as follows: All solutions to C are solution to the query, or if only a yes/no answer is expected: If C has a solution, then the answer is yes, otherwise the answer is no. Such an approach is proposed in [Höhfeld and Smolka, 1988, Bläsius and Hedtstück, 1988, Jaffar and Lassez, 1986].

Acknowledgement
I would like to thank Jochen Doerre and Gert Smolka for discussion that contributed to the paper.

References

[Aït-Kaci, 1984] Hassan Aït-Kaci, A lattice theoretic approach to computation based on a calculus of partially ordered type structures, PhD Dissertation, Univ. of Pennsylvania, 1984

[Aït-Kaci and Nasr, 1986] Hassan Aït-Kaci, Roger Nasr, LOGIN: A Logic Programming Language with built-in inheritance. J. Logic Programming 3: 185-215, (1986)

[Bläsius and Hedtstück, 1988] Karl-Hans Bläsius, Ulrich Hedtstück, Resolution with feature unification, Lecture Notes in Computer Science 329: 17-26, (1988)

[Boone, 1959], W.W. Boone, The word problem, Ann. Math. (2), 70, 207-265, (1959)

[Brachman and Levesque, 1984] Ronald J. Brachman, Hector J. Levesque, The tractability of subsumption in frame-based description languages, Proceedings AAAI-84, pages 34-37, Austin, TX, 1984

[Brachman and Schmolze, 1985] Ronald J. Brachman, James G. Schmolze, An overview of the KL-ONE knowledge representation system. Cognitive Science 9(2): 171-216, 1985

[Brachman et. al., 1983] Ronald J. Brachman, Richard E. Fikes, Hector J. Levesque, Krypton: Integrating Terminology and Assertion, Proceedings AAAI-83, pages 31-35, Washington DC, 1983

[Brachman et. al., 1985] Ronald J.Brachman, Victoria P. Gilbert, Hector J. Levesque, An essential hybrid reasoning system: Knowledge and symbol level accounts of KRYPTON, Proceedings of the Ninth International Joint Conference on Artificial Intelligence 1985, pages 532-539, Los Angeles, 1985

[Burris and Sankappanavar, 1981] Stanley Burris, H.P. Sankappanavar, A course in universal algebra, Springer-Verlag, 1981

[Grätzer, 1979] George Grätzer, Universal Algebra, Springer-Verlag, 1979

[Höhfeld and Smolka, 1988], Markus Höhfeld, Gert Smolka, Definite relations over constraint languages, LILOG report 53, IBM Deutschland, West Germany, 1988

[Jaffar and Lassez, 1986], J. Jaffar, J.-L. Lassez, Constraint Logic Programming,

4th IEEE Symposium on Logic Programming, San Francisco, 1986
[Kaczmarek et. al. 1986] Thomas S. Kaczmarek, Raymond Bates, Gabriel Robins, Recent developments in NIKL, Proceedings AAAI-86, pages 978-985, 1986
[Kay, 1985], M. Kay, Parsing in functional unification grammar, In D. Dowty, L. Kartunnen and A. Zwicky (Eds.), Natural language parsing, Cambridge University Press, 1985
[Levesque and Brachman, 1985] Hector J. Levesque, Ronald J. Brachman, A fundamental tradeoff in knowledge representation, In Readings in Knowledge Representation, Morgan Kaufmann, 1985
[Levesque and Brachman, 1987] Hector J. Levesque, Ronald J. Brachman, Expressiveness and tractability in knowledge representation and reasoning, Comput. Intell. 3, 78-93, (1987)
[von Luck et. al., 1987] Kai von Luck, Bernhard Nebel, Christof Peltason C., Albrecht Schmiedel, The anatomy of the BACK system. KIT-report 41, Technische Universität Berlin, F.R. Germany, 1987
[Mac Gregor and Bates, 1987] Robert M. Mac Gregor, Raymond Bates, The Loom knowledge representation language, Technical report ISI/RS-87-188, Information Sciences Institute, Univ. of Southern California, 1987
[Mac Gregor, 1987] Robert M. Mac Gregor, A deductive pattern matcher, Proceedings AAAI-88, (1988)
[Nebel, 1988], Bernhard Nebel, Computational complexity of terminological reasoning in BACK, Artifical Intelligence 34, 371-383, (1988)
[Novikov, 1955] P.S.Novikov,On the algorithmic undecidability of the word problem in group theory, Trudy Mat. Inst. Steklov. 14, Izdat. Nauk SSSR, Moscow, 1955
[Patel-Schneider, 1984] Peter F. Patel-Schneider, Small can be beautiful in knowledge representation, Proceedings IEEE workshop on principles of knowledge based systems, pages 11-16, Denver, Colorado, 1984
[Patel-Schneider, 1989] Peter F. Patel-Schneider, Undecidability of subsumption in NIKL, (to appear in Artificial Intelligence), 1989
[Patel-Schneider, 1987a] Peter F. Patel-Schneider, Decidable, logic-based knowledge representation, PhD thesis, Dept. of Comp. Science, Univ. of Toronto, 1987
also: Technical report 56, Schlumberger Palo Alto Research, 1987,
also: Tech. report 201/87, Dept. of Comp. Science, Univ. of Toronto, 1987
[Patel-Schneider, 1987b] Peter F. Patel-Schneider, A decidable first-order logic for knowledge representation, draft, submitted to JAR, 1987
[Schild, 1988] Klaus Schild, Undecidability of subsumption in U, draft, Institut für Software und theoretische Informatik, Technische Universität Berlin, 1988
[Schmidt-Schauß and Smolka, 1988] Manfred Schmidt-Schauß, Gert Smolka, Attributive concept descriptions with unions and complements, SEKI-report SR-88-21, Universität Kaiserslautern, (1988)
[Schmolze and Israel, 1983], J.G. Schmolze, D.J. Israel, KL-ONE: Semantics and Classification. in: C.L. Sidner, Research in knowledge representation and

natural language understanding, BBN Technical report 5421, Pages 27-39, Bolt, Beranek and Newman, Cambridge, MA, 1983

[Shieber, 1986] Stuart M. Shieber, An introduction to unification-based approaches to grammar, CSLI Lecture Notes 4, Stanford University, 1986

[Smolka and Aït-Kaci, 1987] Gert Smolka, Hassan Aït-Kaci, Inheritance hierarchies: Semantics and Unification. MCC Report AI-057-87, MCC, Austin, Texas, 1987

[Smolka, 1988], Gert Smolka, A feature logic with subsorts, LILOG report 33, IBM Deutschland, Stuttgart, West Germany, 1988

[Stillwell, 1982], J. Stillwell, The word problem and the isomorphism problem for groups, Bulletin AMS 6(1):33-56, 1982

[Vilain, 1985], Marc Vilain, The restricted language architecture of a hybrid representation system, Proceedings of the Ninth International Joint Conference on Artificial Intelligence 1985, pages 547-551, Los Angeles, 1985