Decidable Reasoning in Terminological Knowledge Representation Systems*

Martin Buchheit+
Deutsches Forschungszentrum fur KI (DFK1)
Stuhlsatzenhausweg 3
D-6600 Saarbriicken, Germany
e-mail: buchheit@dfki.uni-sb.de

Francesco M. **Donini, Andrea Schaerf**
Dipartimento di Informatica e Sistemistica
Universita di Roma "La Sapienza"
Via Salaria 113, 1-00198 Roma, Italy
e-mail: {donini,aschaerf}@assi.ing.uniromal.it

Abstract

Terminological Knowledge Representation Systems (TKRSs) are tools for designing and using knowledge bases that make use of terminological languages (or concept languages). The TKRS we consider in this paper is of practical interest since it goes beyond the capabilities of presently available TKRSs. First, our TKRS is equipped with a highly expressive concept, language, called ALCNR, including general complements of concepts, number restrictions and role conjunction. Second, it allows one to express inclusion statements between general concepts, in particular to express terminological cycles. We provide a sound, complete and terminating calculus for reasoning in ALCNR-knowledge bases based on the general technique of constraint systems.

1 Introduction

A general characteristic of many proposed Terminological Knowledge Representation Systems (TKRSs) such as BACK, LOOM, CLASSIC, KRIS, [Rich, 1991: Woods and Schmolze, 1992] is that they are made up of two different components. Informally speaking, the first is a general schema concerning the classes of individuals to be represented, their general properties and mutual relationships, while the second is a (partial) instantiation of this schema, containing assertions relating either individuals to classes, or individuals to each other.

Retrieving information in actual knowledge bases (KBs) built up using one of these systems is a deductive process involving both the schema (TBox) and its instantiation (ABox).

During the realization and use of a KB. a TKRS should provide a mechanical solution for at least the following problems (from now on, we use the word concepts to refer to classes):

1. Concept Satisfiability: given a KB and a concept C, does there exist at least one model of the KB assigning a nonempty extension to C?
2 Subsumption. given a KB and two concepts C and D, is C more general than D in any model of the KB?
3. KB-saiisftability, are an ABox and a TBox consistent with each other?
4. Instance Checking: given a KB, an individual a and a concept C, is a an instance of C in any model of the KB[7]

Up to now, all the proposed systems (except for KRIS) give incomplete procedures for solving the above problems 1 4. That is, some inferences are missed, in some cases without a precise semantical characterization of which ones are. If the designer or the user needs a (more) complete reasoning, she/he must either write programs in a suitable programming language, or define appropriate inference rules completing the inference capabilities of the system (as in BACK, LOOM, and CLASSIC).

In our opinion incomplete procedures are just a provisional answer to the problem—the best possible up to now. In order to improve on such an answer, a theoretical analysis of the general problems 1-4 must be done. But most importantly, theoretical analysis is needed for making cyclic definitions of concepts (see [Nebel, 1990, Chapter 5]) fully available in TKRSs. Such a feature is of undoubtable practical interest, yet present TKRSs can only approximate cycles, by using forward inference rules.

Previous results do not deal with the problems 1-4 in their full generality. The problems are studied in [Nebel, 1990, Chapter 4], but only incomplete procedures are given, and cycles are not **considered.** In [Donini et a/., 1992] the complexity of instance checking has been analyzed, but only KBs without a TBox are treated. Previous theoretical work on cycles **was** done in [Baader, 1990b; Baader, 1990a; Nebel, 1990; Nebel, 1991; Schild, 1991], but only **KBs** formed by the I Box alone are considered. Moreover, **these approaches** do not deal with number restrictions (except **for** [Nebel,

*This work has been supported by the ESPRIT Basic Research Action N.6810 COMPULOG 2 and by the Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo of the CNR (Italian Research Council).

+This research was partly done while the first author was visiting the Dipartimento di Informatica e Sistemistica. Universita di Roma "La Sapienza".

1990, Section 5.3.5]), which constitute a basic feature already provided by many TKRSs, and the techniques used do not seem easily extensible to reasoning with A Boxes.

In this paper, we propose a TKRS equipped with a highly expressive language of practical significance, and prove the decidability of problems 1-4. In particular, our system makes use of the language *ALCNR.,* which supports general complements of concepts, number restrictions and role conjunction. Moreover, the system allows one to express inclusion statements between general concepts and, as a particular case, terminological cycles We prove decidability by means of a suitable calculus, which is developed within the quite well established framework of constraint systems (see [Donini *et al.,* 1991a: Schmidt-SchauB and Smolka, 1991]) thus exploiting a uniform approach to reasoning in TKRSs. Moreover, our calculus can easily be turned into a decision procedure.

The paper is organized as follows. In Section 2 we introduce the language, and we give it a Tarski-style extensional semantics, which is the most commonly used In Section 3 we provide a calculus, and show its correctness and termination. In Section 4 we consider a refinement of our calculus, working in exponential space. In See tion 5 we establish the equivalence of general inclusion statements and general concept definitions using the descriptive semantics. Finally, we discuss in detail several practical impacts of our results in Section 6 For the sake of brevity proofs are omitted. They can be found in [Buchheit *et* al., 1993].

2 Preliminaries

In concept languages, concepts represent the classes of objects in the domain of interest, while roles represent binary relations between objects. Complex concepts and roles can be defined by means of suitable constructors applied to primitive concepts and primitive roles. In particular, concepts and roles in *ALCN'R* can be formed by means of the following syntax *(A* denotes a primitive concept, P_i *(i* = 1, . . . , m*)* denotes a primitive role. *C* and *D* denote arbitrary concepts and *R* an arbitrary role).

C, D	—	$A \mid$	(primitive concept)
		$\top \mid$	(top)
		$\bot \mid$	(bottom)
		$(C \sqcap D) \mid$	(conjunction)
		$(C \sqcup D) \mid$	(disjunction)
		$\neg C \mid$	(complement)
		$\forall R.C \mid$	(universal quantification)
		$\exists R.C \mid$	(existential quantification)
		$(\geq n\ R) \mid (\leq n\ R)$	(number restrictions)
R	—	$P_1 \sqcap \cdots \sqcap P_m$	(role conjunction)

We interpret concepts as subsets of a domain and roles as binary relations over a domain. More precisely, an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a nonempty set $\Delta^{\mathcal{I}}$ (the *domain* of $\mathcal{I}$) and a function $\cdot^{\mathcal{I}}$ (the *extension function* of $\mathcal{I}$) that maps every concept to a subset of $\Delta^{\mathcal{I}}$ and every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that the following equations are satisfied ($\sharp\{\}$ denotes the cardinality of a set):

$$
\begin{aligned}
&\top^{\mathcal{I}} = \Delta^{\mathcal{I}},\ \bot^{\mathcal{I}} = \emptyset,\ (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}},\\
&(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}},\ (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}\\
&(\forall R.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \forall e : (d,e) \in R^{\mathcal{I}} \rightarrow e \in C^{\mathcal{I}}\}\\
&(\exists R.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists e : (d,e) \in R^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}\\
&(\geq n\ R)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid\ \ \sharp\{e \mid (d,e) \in R^{\mathcal{I}}\} \geq n\}\\
&(\leq n\ R)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid\ \ \sharp\{e \mid (d,e) \in R^{\mathcal{I}}\} \leq n\}\\
&(P_1 \sqcap \cdots \sqcap P_m)^{\mathcal{I}} = P_1^{\mathcal{I}} \cap \cdots \cap P_m^{\mathcal{I}}
\end{aligned}
$$

A KB built by means of concept languages is formed by two components: The *intenstonal* one, called the TBox, and the *extensional* one, called the ABox.

We first turn our attention to the TBox. As we said before, the intensional level specifies the properties of the concepts of interest in a particular application. Syntactically, such properties are expressed in terms of so-called *inclusion statements* (see [Nebel, 1990, Chapter 3]). An inclusion statement (or simply inclusion) has the form

$$C \sqsubseteq D$$

where C and D are two arbitrary concepts. Intuitively, the statement specifies that every instance of C is also an instance of D. More precisely, an interpretation $\mathcal{I}$ *satisfies* the inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. A TBox $\mathcal{T}$ is a finite set of inclusions. An interpretation $\mathcal{I}$ is a *model* for a TBox $\mathcal{T}$ if $\mathcal{I}$ satisfies all inclusions in $\mathcal{T}$.

Many TKRSs provide the user with mechanisms for stating *concept definitions* (e.g. [Nebel, 1990, Section 3.2]) of the form $A \doteq D$ (interpreted as set equality), or $A \dot{\subseteq} D$ (primitive concept definition, interpreted as set inclusion), with the restrictions that the left-hand side concept A must be a concept name, that for each concept name at most one definition is allowed, and that no so-called *terminological cycles* are allowed, i.e. no concept name may occur—neither directly nor indirectly—within its own definition.

We do not impose any of these restrictions to the form of inclusions, obtaining statements that are syntactically more expressive than concept definitions. In particular, a definition of the form $A \doteq D$ can be expressed in our system using the pair of inclusions $A \sqsubseteq D$ and $D \sqsubseteq A$, whereas an inclusion of the form $C \sqsubseteq D$, where C and D are arbitrary concepts, cannot be expressed with concept definitions. Moreover, cyclic inclusions are allowed in our statements, realizing terminological cycles.

As shown in [Nebel, 1991], there are at least three types of semantics for terminological cycles, namely the least fixed point, the greatest fixed point, and the descriptive semantics. However, fixed point semantics apply only to fixed point statements, which are much less general than our inclusion statements. Instead, the descriptive semantics interprets statements as just restricting the set of possible models, with no definitional import. Hence, it can be suitably extended to our case, and is exactly the one we adopt.

We can now turn our attention to the *extensional level* of the KB, i.e. the ABox. The ABox essentially allows one to specify instance-of relations between individuals and concepts, and between pairs of individuals and roles.

Let $\mathcal{O}$ be an alphabet of symbols, called *individuals*. Instance-of relationships are expressed in terms of *membership assertions* of the form:

$$C(a), \qquad R(a,b)$$

where a and b are individuals, C is a concept, and R is a role. Intuitively, the first form states that a is an instance of C, whereas the second form states that a is related to b by means of the role R.

In order to assign a meaning to membership assertions, the extension function $\cdot^{\mathcal{I}}$ of an interpretation $\mathcal{I}$ is extended to individuals by mapping them to elements of $\Delta^{\mathcal{I}}$ in such a way that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$ (Unique Name Assumption). An interpretation $\mathcal{I}$ *satisfies* the assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and *satisfies* $R(a,b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. An ABox $\mathcal{A}$ is a finite set of membership assertions. $\mathcal{I}$ is a *model* for an ABox $\mathcal{A}$ if $\mathcal{I}$ satisfies all the assertions in $\mathcal{A}$.

An *$\mathcal{ALCNR}$-knowledge base* Σ is a pair $\Sigma = (\mathcal{T}, \mathcal{A})$ where $\mathcal{T}$ is a TBox, and $\mathcal{A}$ is an ABox. An interpretation $\mathcal{I}$ is a *model* for Σ if it is both a model for $\mathcal{T}$ and a model for $\mathcal{A}$. We can now formally define the problems 1-4 mentioned in the introduction, w.r.t. a given KB Σ:

1. *Concept Satisfiability*: C is *satisfiable* w.r.t. Σ, if there exists a model $\mathcal{I}$ of Σ such that $C^{\mathcal{I}} \neq \emptyset$;
2. *Subsumption*: C is *subsumed* by D w.r.t. Σ, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of Σ;
3. *KB-satisfiability*: Σ itself is *satisfiable*, if it has a model;
4. *Instance Checking*: a is an instance of C, written $\Sigma \models C(a)$, if the assertion $C(a)$ is satisfied in every model of Σ.

Example 2.1 Consider the following KB $\Sigma = (\mathcal{T}, \mathcal{A})$:

$\mathcal{T} = \{$ $\exists$TEACHES.Course $\sqsubseteq$ (Student $\sqcap$ $\exists$DEGREE.BS) $\sqcup$ Prof,
Prof $\sqsubseteq$ $\exists$DEGREE.MS, $\exists$DEGREE.MS $\sqsubseteq$ $\exists$DEGREE.BS,
MS $\sqcap$ BS $\sqsubseteq$ $\bot$ $\}$

$\mathcal{A} = \{$TEACHES(john,cs1), ($\leq$ 1 DEGREE)(john), Course(cs1)$\}$

It is easy to see that Σ is satisfiable. Notice also that it is possible to draw several non-trivial conclusions from Σ. For example, we can infer that $\Sigma \models$ Student(john). Intuitively this can be shown as follows: john teaches a course, thus he is either a student with a BS or a professor. But he can't be a professor since professors have at least two degrees (BS and MS) and he has at most one, therefore he is a student. □

Given the above semantics, the problems 1-4 can all be reduced to KB-satisfiability (or to its complement) in linear time. In fact, given a KB $\Sigma = (\mathcal{T}, \mathcal{A})$, a concept C is satisfiable w.r.t. Σ iff the KB $(\mathcal{T}, \mathcal{A} \cup \{C(b)\})$ is satisfiable, and C is subsumed by D w.r.t. Σ iff the KB $(\mathcal{T}, \mathcal{A} \cup \{(C \sqcap \neg D)(b)\})$ is not satisfiable, where b is a new individual not appearing in Σ. Finally, $\Sigma \models C(a)$ iff the KB $(\mathcal{T}, \mathcal{A} \cup \{(\neg C)(a)\})$ is not satisfiable. Hence, we can concentrate just on KB-satisfiability in the next section.

3 Decidability Result

In this section we present a calculus for deciding KB-satisfiability and state its correctness and termination.

Our method makes use of the notion of constraint systems [Donini *et al.*, 1991a; Schmidt-Schauß and Smolka, 1991; Donini *et al.*, 1991b], and is based on a tableaux-like calculus that tries to build a model for the logical formula corresponding to a KB.

Consider an alphabet of variable symbols $\mathcal{V}$. The elements of $\mathcal{V}$ are denoted by the letters x, y, z, w. In the sequel we use the term *object* as an abstraction for individual and variable (i.e. an object is an element of $\mathcal{O} \cup \mathcal{V}$). Objects are denoted by the symbols s, t and, as in Section 2, individuals are denoted by a, b. In the rest of this section, R denotes the role $R = P_1 \sqcap \ldots \sqcap P_k$ $(k \geq 1)$.

A *constraint* is a syntactic entity of one of the forms:

$$s{:}C, \quad sPt, \quad \forall x.x{:}C, \quad s \neq t,$$

where C is a concept and P is a primitive role. Concepts are assumed to be *simple*, i.e. they contain only complements of the form $\neg A$, where A is a primitive concept. Arbitrary $\mathcal{ALCNR}$-concepts can be rewritten into equivalent simple concepts in linear time [Donini *et al.*, 1991a]. A *constraint system* is a finite nonempty set of constraints.

Given an interpretation $\mathcal{I}$, we define an *$\mathcal{I}$-assignment* α as a function that maps every variable in $\mathcal{V}$ to an element of $\Delta^{\mathcal{I}}$ (not necessarily injectively), and every individual a to $a^{\mathcal{I}}$ (i.e. $\alpha(a) = a^{\mathcal{I}}$ for all $a \in \mathcal{O}$).

A pair $(\mathcal{I}, \alpha)$ *satisfies* the constraint $s{:}C$ if $\alpha(s) \in C^{\mathcal{I}}$, the constraint sPt if $(\alpha(s), \alpha(t)) \in P^{\mathcal{I}}$, the constraint $s \neq t$ if $\alpha(s) \neq \alpha(t)$, and finally, the constraint $\forall x.x{:}C$ if $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$. A constraint system S is *satisfiable* if there is a pair $(\mathcal{I}, \alpha)$ that satisfies every constraint in S.

An $\mathcal{ALCNR}$-knowledge base $\Sigma = (\mathcal{T}, \mathcal{A})$ can be translated into a constraint system S_Σ by replacing every inclusion $C \sqsubseteq D \in \mathcal{T}$ with the constraint $\forall x.x{:}\neg C \sqcup D$, every membership assertion $C(a)$ with $a{:}C$, $R(a,b)$ with $aP_1b, \ldots, aP_kb$, and including the constraint $a \neq b$ for every pair (a,b) of individuals appearing in $\mathcal{A}$. It is easy to see that Σ is satisfiable if and only if S_Σ is satisfiable.

In order to check a constraint system S for satisfiability, our technique adds constraints to S until either an evident contradiction is generated or an interpretation satisfying it can be obtained from the resulting system. Constraints are added on the basis of a suitable set of so-called *propagation rules*.

Before providing the rules, we need some additional definitions. Let S be a constraint system. We say that t is an *R-successor of s in S* if $sP_1t, \ldots, sP_kt$ are in S. We say that t is a *1-successor of s in S* if for some role R, t is an R-successor of s. If S is clear from the context we simply say that t is an R-successor or a 1-successor of s. Moreover, we denote by *successor* the transitive closure of the relation 1-successor, and we denote by *predecessor* its inverse.

We denote by $S[x/s]$ the constraint system obtained from S by replacing each occurrence of the variable x by s. We say that s and t are *separated in S* if the constraint $s \neq t$ is in S.

Given a constraint system S we say that two variables x and y are *S-equivalent*, written $x \equiv_S y$, if $\{C \mid x{:}C \in S\} = \{C \mid y{:}C \in S\}$. Intuitively, two S-equivalent variables could represent the same element in the potential interpretation built by the rules, unless they were separated.

The *propagation rules* are:

$S \rightarrow_{\sqcap}$ $\{s:C_1,\ s:C_2\} \cup S$

if $s:C_1 \sqcap C_2$ is in S, $s:C_1$ and $s:C_2$ are not both in S

$S \rightarrow_{\sqcup}$ $\{s:D\} \cup S$

if $s:C_1 \sqcup C_2$ is in S, neither $s:C_1$ nor $s:C_2$ is in S and $D = C_1$ or $D = C_2$

$S \rightarrow_{\forall}$ $\{t:C\} \cup S$

if $s:\forall R.C$ is in S, t is a R-successor of s, $t:C$ is not in S

$S \rightarrow_{\exists}$ $\{sP_1y, \ldots, sP_ky,\ y:C\} \cup S$

if $s:\exists R.C$ is in S, y is a new variable, there is no t such that t is a R-successor of s in S and $t:C$ is in S; if s is a variable there is no variable w in S such that w is a predecessor of s and $s \equiv_S w$

$S \rightarrow_{\geq}$ $\{sP_1y_i, \ldots, sP_ky_i \mid i \in 1..n\} \cup$
$\{y_i \neq y_j \mid i,j \in 1..n, i \neq j\} \cup S$

if $s:(\geq n\ R)$ is in S, $y_1, \ldots, y_n$ are new variables, there do not exist n pairwise separated R-successors of s in S; if s is a variable there is no variable w such that w is a predecessor of s and $s \not\equiv_S w$

$S \rightarrow_{\leq}$ $S[y/t]$

if $s:(\leq n\ R)$ is in S, s has more than n R-successors in S, y, t are two R-successors of s that are not separated

$S \rightarrow_{\forall x}$ $\{s:C\} \cup S$

if $\forall x.x:C$ is in S, s appears in S, $s:C$ is not in S.

We call the rules $\rightarrow_{\sqcup}$, $\rightarrow_{\leq}$ *nondeterministic* rules, since they can be applied in different ways to the same constraint system. All the other rules are called *deterministic* rules. Moreover, we call the rules $\rightarrow_{\exists}$, $\rightarrow_{\geq}$ *generating* rules, since they introduce new variables in the constraint system. All other rules are called *nongenerating* ones.

The use of the condition based on the S-equivalence relation in the generating rules is related to the goal of keeping the constraint system finite even in presence of potentially infinite chains of applications of generating rules.

One can verify that rules are always applied to a system S either because of the presence in S of a given constraint $s:C$ or, in the case of the $\rightarrow_{\forall x}$-rule, because of the presence of an object s in S. When no confusion arises, we will say that a rule is *applied to* the object s.

Proposition 3.1 (Invariance) *Let S and S' be constraint systems. Then:*

1. *If S' is obtained from S by application of a deterministic rule, then S is satisfiable if and only if S' is satisfiable.*
2. *If S' is obtained from S by application of a nondeterministic rule, then S is satisfiable if S' is satisfiable. Furthermore, if a nondeterministic rule applies to S, then it can be applied in such a way that it yields a constraint system S' which is satisfiable if and only if S is satisfiable.*

Given a constraint system, more than one rule might be applicable to it. We define the following *strategy* for the application of rules:

1. apply a rule to a variable only if no rule is applicable to individuals;
2. apply generating rules only if no nongenerating rule is applicable;
3. apply a generating rule to a variable x only if no rule is applicable to a predecessor of x.

A constraint system is *complete* if no propagation rule applies to it. A complete system derived from a system S is also called a *completion* of S. A *clash* is a constraint system having one of the following forms:

- $\{s:\bot\}$
- $\{s:A,\ s:\neg A\}$, where A is a primitive concept.
- $\{s:(\leq n\ R)\} \cup \{sP_1t_i, \ldots, sP_kt_i \mid i \in 1..n+1\}$
  $\cup \{t_i \neq t_j \mid i,j \in 1..n+1, i \neq j\}$.

In order to obtain an interpretation from a complete constraint system we need some additional definitions. Let S be a constraint system and x, w be variables in S. We call w a *witness of x in S* if the three following conditions hold:

1. $x \equiv_S w$
2. w is a predecessor of x in S
3. no rule is applicable to any predecessor of x.

Notice that the third condition ensures that no new constraint will be imposed on x. We say *x is blocked (by w)* if x has a witness (w) in S. In a constraint system obtained from an S_Σ by applying the rules according to the strategy, a blocked variable has exactly one witness and no successors.

Let S be a constraint system. We define the *canonical interpretation $\mathcal{I}_S$* for S and the *canonical $\mathcal{I}_S$-assignment* α_S for S as follows:

1. $\Delta^{\mathcal{I}_S} := \{s \mid s \text{ is an object in } S\}$
2. $\alpha_S(s) := s$
3. $s \in A^{\mathcal{I}_S}$ iff $s:A$ is in S
4. $(s,t) \in P^{\mathcal{I}_S}$ iff
   (a) sPt is in S or
   (b) s is a blocked variable, w is the witness of s in S and wPt is in S.

The canonical interpretation of complex concepts and roles can be uniquely reconstructed from the interpretations of the primitive ones imposing the equations stated in Section 2.

Theorem 3.2 (Correctness) *Let S be a complete constraint system. S is satisfiable iff it contains no clash.*

Proof. (Sketch) If S contains a clash, it is clearly unsatisfiable. If S contains no clash, it can be shown that the pair $(\mathcal{I}_S, \alpha_S)$ satisfies every constraint in S. □

The termination of the calculus is based on the following property: In any constraint system obtained from an S_Σ by applying the rules according to the strategy, every variable can have at most 2^n variables among its predecessors, where n is the size of Σ. The technique we use is similar to the standard *filtration* technique used in modal logics, with the main difference that we must take into account separated variables.

Theorem 3.3 (Termination) *Let S be a constraint system. Every completion of S is finite.*

Notice that, since the domain of the canonical interpretation $\Delta^{\mathcal{I}_S}$ is always finite, we have also implicitly proved that $\mathcal{ALCNR}$-knowledge bases have the *finite model property*, i.e. any satisfiable knowledge base has a finite model.

Theorem 3.4 (Decidability) *Checking whether an $\mathcal{ALCNR}$-KB is satisfiable is a decidable problem.*

4 A Calculus Working in Exponential Space

The calculus proposed in the previous section requires to compute all the completions of the constraint system S_Σ. Unfortunately, such completions may be of double exponential size w.r.t. the size of Σ.

For an exponential space algorithm it is therefore crucial not to keep an entire complete constraint system in memory, but to store only small portions at a time.

We give propagation rules, called *trace rules*, that build up only a portion of complete constraint systems.

The *trace rules* consist of the $\rightarrow_{\sqcap}$-, $\rightarrow_{\sqcup}$-, $\rightarrow_{\forall}$-, $\rightarrow_{\forall_r}$- and the $\rightarrow_{\leq}$-rule (the nongenerating rules) together with the two generating rules $\rightarrow_{T\exists}$ and $\rightarrow_{T\geq}$, which are obtained respectively from the $\rightarrow_{\exists}$- and the $\rightarrow_{\geq}$-rule adding the following condition of application:

for all constraints tPx in S, either t is a predecessor of s or $t = s$.

Let T be a constraint system obtained from S_Σ by application of the trace rules. We call T a *trace* of S_Σ if no trace rule applies to T.

If the trace rules are applied according to the strategy they show the following behavior: Given an object s, if at least one generating rule is applicable, all its 1-successors $y_1, \ldots, y_n$ are introduced. Then, after nongenerating rules are applied, one variable y_i is (nondeterministically) chosen, and all 1-successors of y_i are introduced. Unlike normal propagation rules, trace rules introduce no successor for any object different from y_i. Then, one variable is chosen among the 1-successors of y_i, only its 1-successors are added to the constraint system, and so on.

The reason why we introduce all the 1-successors of the "chosen" object is the following. For every chosen object s all 1-successors of s must be present simultaneously at some stage of the computation, since only the interplay of role conjunction and number restrictions forces us to identify certain successors. This is important because, when identifying variables, the constraints imposed on them are combined, which may lead to clashes that otherwise would not have occurred.

A calculus based on trace rules, for deciding satisfiability and subsumption of $\mathcal{ALCNR}$-concepts, was previously defined in [Donini *et al.*, 1991a]. Algorithms exploiting traces have been given in [Hollunder *et al.*, 1990]. However, all these works do not deal with KBs, but only with concept expressions.

Proposition 4.1 *Let Σ be an $\mathcal{ALCNR}$-KB, S_Σ its associated constraint system, and let n be the size of Σ. Then:*

1. *The length of a trace rule derivation issuing from S_Σ is bounded by 2^n.*
2. *Every complete constraint system extending S_Σ can be obtained as the union of finitely many traces.*
3. *Suppose S is a complete constraint system extending S_Σ and $\mathcal{T}$ is a finite set of traces such that $S = \bigcup_{T \in \mathcal{T}} T$. Then S contains a clash if and only if some $T \in \mathcal{T}$ contains a clash.*

Since an $\mathcal{ALCNR}$-KB Σ is satisfiable if and only if there exists a complete constraint system derivable from S_Σ without a clash, it follows from Proposition 4.1 that satisfiability of an $\mathcal{ALCNR}$-KB can be decided with exponential space. A possible algorithm using space $2^{p(n)}$ where $p(n)$ is a polynomial in the size of S_Σ may be the following: compute one complete constraint system, one trace at a time, trying all possible applications of the nondeterministic rules $\rightarrow_{\leq}$, $\rightarrow_{\sqcup}$, and finding the choices leading to traces without a clash.

Theorem 4.2 *Satisfiability of an $\mathcal{ALCNR}$-KB can be decided with exponential space.*

Thanks to an unpublished manuscript by D. McAllester, and (independently) from an observation by W. Nutt, we know that deciding the satisfiability of an $\mathcal{ALCNR}$-KB is an EXPTIME-hard problem. Hence, we do not expect to find any algorithm solving the problem in polynomial space, unless PSPACE=EXPTIME. Nevertheless, the algorithm outlined above may require double exponential time. It is still open whether there exists an algorithm working in (simple) exponential time.

5 Inclusions versus Concept Definitions

In this section we compare the expressive power of TBoxes defined as a set of inclusions (as done in this paper) and TBoxes defined as a set of (possibly cyclic) concept definitions of the form $A \dot{\leq} D$ and $A \doteq D$.

Unlike [Baader, 1990a] and [Schild, 1991], we consider reasoning problems dealing with TBox and ABox together. Moreover, we use the descriptive semantics for the concept definitions, as we do for the inclusions. The result we have obtained is that inclusion statements and concept definitions actually have the same expressive power. In details, we show that the satisfiability of a knowledge base $\Sigma = (\mathcal{A}, \mathcal{T})$, where $\mathcal{T}$ is a set of inclusion statements, can be reduced to the satisfiability of a knowledge base $\Sigma' = (\mathcal{A}', \mathcal{T}')$ such that $\mathcal{T}'$ is a set of concept definitions. The other direction—from concept definitions to inclusions—is trivial since definitions can be expressed by pairs of inclusions (see Section 2).

As a notation, given a TBox $\mathcal{T} = \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\}$, we define the concept $C_\mathcal{T}$ as $C_\mathcal{T} = (\neg C_1 \sqcup D_1) \sqcap \cdots \sqcap (\neg C_n \sqcup D_n)$. As pointed out in [Baader, 1990a] for $\mathcal{ALC}$, an interpretation satisfies a TBox $\mathcal{T}$ iff it satisfies the equation $C_\mathcal{T} = \top$. This result easily extends to $\mathcal{ALCNR}$.

Given a knowledge base $\Sigma = (\mathcal{A}, \mathcal{T})$ and a concept A not appearing in Σ, we define the knowledge base $\Sigma' =$

$(\mathcal{A}', \mathcal{T}')$ as follows:

$$\begin{aligned} \mathcal{A}' &= \mathcal{A} \cup \{A(b) \mid b \text{ is an individual in } \Sigma\} \\ \mathcal{T}' &= \{A \dot{\leq} C_{\mathcal{T}} \sqcap \forall P_1.A \sqcap \cdots \sqcap \forall P_n.A\} \end{aligned}$$

where $P_1, P_2, \ldots, P_n$ are all the primitive roles appearing in Σ. Interpreting the primitive concept definition ($\dot{\leq}$) as an inclusion, and exploiting canonical interpretations, we are able to prove the following result.

Theorem 5.1 *Σ is satisfiable iff Σ' is satisfiable.*

6 Discussion

In this paper we have proved the decidability of the main inference services of a TKRS based on the concept language *ALCNR.* We believe that this result is not only of theoretical importance, but has the following impacts on existing TKRSs.

First of all, a complete procedure working in exponential space can be easily devised from the calculus provided in Sections 3 and 4. From this procedure, one can build more efficient (but still complete) ones by rippling optimization techniques. Such procedures might work well in practical cases, despite their worst case intractability.

Secondly, a complete procedure (possibly optimized) offers a benchmark for comparing incomplete proredures, not only in terms of performance, but also in terms of missed inferences. In fact, incomplete procedures can be meaningfully compared only if missed inferences are considered. However, to recognize missed inferences over large examples, one needs exactly a complete procedure—even if not an efficient one like ours

Thirdly, new incomplete procedures can be obtained from the calculus by modifying some of the propagation rules. Since the rules build up a model, modifications to them have a semantical counterpart which give* a precise account, of the incomplete procedures obtained. For instance, define the depth of a variable x as the number of variables which are predecessors of *x* Then, an incomplete calculus could be devised, which generates variables only to a given depth—say, linear depth in the size of the KB. This calculus would miss contradictions (and hence inferences, by refutation) occurring in variables which are "far away" from the known individuals of the KB, and this is a meaningful explanation of the incompleteness, even for a non-expert user. From a com putational point of view, an immediate consequence of the complexity analysis carried over in this paper is that such an incomplete procedure would run in polynomial space.

Acknowledgements

Maurizio Lenzerini inspired us to this work and contributed to the paper in several discussions. Franz Baader and Werner Nutt made many helpful comments on earlier drafts. Scott Benson helped us in improving the English. The third author also acknowledges Yoav Shoham for his hospitality at the Computer Science Department of Stanford University.

References

[Baader, 1990a] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. Technical Report RR-90-13, DFKI, Kaiserslautern, 1990. A shorter version appeared in *Proc. of IJCAI-91,* pages 446-451, 1991.

[Baader. 1990b] F. Baader. Terminological cycles in KL-ONE-based KR-languages. In *Proc. of AAAI-90,* pages 621-626, 1990.

[Buchheit *et* al., 1993] M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. Technical Report RR-93-10, DFKI, Saarbriicken, 1993.

[Donini *et* al., 1991a] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proc. of KR-91,* pages 151-162. Morgan Kaufniann, 1991.

[Donini et *al..* 1991b] F. M. Donini, M. Lenzerini, D Nardi, and A. Schaerf. A hybrid system integrating datalog and concept languages. In *Proc. of the 2nd Italian Couf. on Artificial Intelligence, number 549 in* LNA1. Springer Verlag, 1991. An extended version appeared also in the Working Notes of the AAAI Fall Symposium 'Principles of Hybrid Reasoning", 1991.

[Donini *et al*., 1992] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. From subsumption to instance checking. Technical Report 15.92, Dipartimento di Informatica e Sistemistica, Universita di Boma "La Sapienza", 1992.

[Holluuder *et al,* 1990] B. Hollunder, W. Nutt, and M Schmidt SchauB. Subsumption algorithms for concept description languages. In *Proc. of ECAI-90,* pages 348-353, London, 1990. Pitman.

[Nebel. 1990] B. Nebel. *Reasoning and Revtston in Hybrid Representation Systems.* Number 422 in LNAI. Springer Verlag, 1990.

[Nebel, 1991] B Nebel. Terminological cycles: Semantics and computational properties. In John F. Sowa, editor, *Principles of Semantic Networks,* pages 331—361. Morgan Kaufmann, 1991.

[Rich. 1991] C Rich, editor. SIGART bulletin. Special issue on implemented knowledge representation and reasoning systems. (2)3, June 1991.

[Schild. 1991] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCA1-91* Sydney, 1991.

[Schnudt-SchauB and Smolka, 199l] M. Schmidt-SchauB and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence,* 48(1):1-26, 1991.

[Woods and Schmolze, 1992] W.A. Woods and J.G. Schmolze. The KL-ONE family. In F.W. Lehmann, editor, *Semantic Networks in Artificial Intelligence,* pages 133 178. Pergamon Press, 1992.