Terminological Cycles in KL-ONE-based Knowledge Representation Languages[1]

Franz Baader

German Research Center for Artificial Intelligence

Projektgruppe WINO, Postfach 2080

D-6750 Kaiserslautern, West Germany

baader@uklirb.uucp

Abstract

Cyclic definitions are often prohibited in terminological knowledge representation languages because, from a theoretical point of view, their semantics is not clear and, from a practical point of view, existing inference algorithms may go astray in the presence of cycles. In this paper, we shall consider terminological cycles in a very small KL-ONE-based language. For this language, the effect of the three types of semantics introduced by (Nebel 1987,1989,1989a) can be completely described with the help of finite automata. These descriptions provide a rather intuitive understanding of terminologies with cyclic definitions and give insight into the essential features of the respective semantics. In addition, one obtains algorithms and complexity results for subsumption determination. As it stands, the greatest fixed-point semantics comes off best. The characterization of this semantics is easy and has an obvious intuitive interpretation. Furthermore, important constructs – such as value-restriction with respect to the transitive or reflexive-transitive closure of a role – can easily be expressed.

1. Introduction

Cyclic definitions are prohibited in most terminological knowledge representation languages (e.g., in KRYPTON (Brachman et al. 1985), NIKL (Kaczmarek et al. 1986), or LOOM (MacGregor & Bates 1987)) for the following reasons. From a theoretical point of view, it is not obvious how to define the semantics of terminological cycles. But even if we have fixed a semantics it is not easy to obtain the corresponding inference algorithms.

On the other hand, cyclic definitions may be very useful and intuitive, e.g., if we want to express the transitive closure of roles (i.e., binary relations). For a role child, value-restrictions with respect to its transitive closure offspring can be expressed by cyclic concept definitions if we take the appropriate semantics. For the same reason, recursive axioms are considered in database research (see e.g., (Aho & Ullman 1979), (Immerman 1982), (Vardi 1982), and (Vielle 1989)). Aho and Ullman have shown that the transitive closure of relations cannot be expressed in the relational calculus, which is a standard relational query language. They proposed to add cyclic definitions which are interpreted by least fixed-point semantics. This was also the starting point for an extensive study of fixed-point extensions of first-order logic (see e.g., (Gurevich & Shelah 1986)).

A thorough investigation of cycles in terminological knowledge representation languages can be found in (Nebel 1987,1989,1989a). Nebel considered three different kinds of semantics – namely, least fixed-point semantics, greatest fixed-point semantics, and what he called descriptive semantics – for cyclic definitions in his language $\mathcal{NTF}$. But, due to the fact that this language is relatively strong[2], he does not provide a deep insight into the meaning of cycles with respect to these three types of semantics. For the two fixed-point semantics, Nebel explicates his point just with a few examples. The meaning of descriptive semantics – which, in Nebel's opinion, comes "closest to the intuitive understanding of terminological cycles" ((Nebel 1989a), p. 124) – is treated more thoroughly. But even in this case the results are not quite satisfactory. For example, the decidability of subsumption determination is proved by an argument[3] which cannot be used to derive a practical algorithm, and which does not give insight into the reason why one concept defined by some cyclic definition subsumes another one.

Before we can determine what kind of semantics is most appropriate for terminological cycles, we should get a better understanding of their meaning. The same argument applies to the decision whether to allow or disallow cycles. Even if cycles are prohibited, this should not just be done because one does not know what they mean and how they can be handled.

In this paper, we shall consider terminological cycles in a very small KL-ONE-based language which allows only concept conjunction and value-restrictions. For this language the effect of the three above mentioned types of semantics can be completely described with the help of finite automata. These descriptions provide a rather intu-

1. This work was supported by the German "Bundesministerium für Forschung und Technologie" under Grant ITW 8903 0.

2. The language allows concept and role conjunction, value-restrictions, number-restrictions and negation of primitive concepts.

3. Roughly speaking, the argument says that it is sufficient to consider only finite interpretations to determine subsumption relations.

itive understanding of terminologies with cyclic definitions and give insight into the essential features of the respective semantics. In addition, subsumption determination for each type of semantics can be reduced to a (more or less) well-known decision problem for finite automata. Hence, existing algorithms can be used to decide subsumption and known complexity results yield the complexity of subsumption determination.

Syntax and (descriptive) semantics of our small terminological language is introduced in the next section. In Section 3, alternative types of semantics – namely least and greatest fixed-point semantics – are considered, which may be more appropriate in the presence of terminological cycles. We shall see that, from a constructive point of view, the greatest fixed-point semantics should be preferred since greatest fixed-point models can be obtained by a single limit process. In Section 4, the three types of semantics are characterized with the help of finite automata. The characterization of the greatest fixed-point semantics is easy and intuitively clear. Subsumption with respect to greatest fixed-point semantics, and – after some modifications of the automaton – also with respect to least fixed-point semantics can be reduced to inclusion of regular languages. For descriptive semantics, we have to consider inclusion of certain languages of infinite words which are defined by the automaton. Fortunately, these languages have already been investigated in the context of monadic second-order logic (see (Büchi 1960)).

2. A Small KL-ONE-based KR-language

In KL-ONE-based languages we start with atomic concepts and roles and can use the language formalism to define new concepts and roles. Concepts can be considered as unary predicates which are interpreted as sets of individuals whereas roles are binary predicates which are interpreted as binary relations between individuals. The languages differ in what kind of constructs are allowed for the definition of concepts and roles. The language considered in this paper has only two constructs, which can be used to define concepts: concept conjunction and value-restriction.

Definition 2.1. (concept terms and terminologies)
Let **C** be a set of concept names and **R** be a set of role names. The set of *concept terms* is inductively defined. As a starting point of the induction,
(1) any element of **C** is a concept term. (atomic terms)
Let C and D be concept terms and let R be a role name.
(2) Then $C \sqcap D$ is a concept term. (concept conjunction)
(3) Then $\forall R{:}C$ is a concept term. (value-restriction)
Let A be a concept name and let D be a concept term. Then A = D is a terminological axiom. A *terminology* (T-box) is a finite set of terminological axioms with the additional restriction that no concept name may appear more than once as a left hand side of a definition. ❑

A T-box contains two different kinds of concept names. *Defined concepts* occur on the left hand side of a terminlogical axiom. The other concepts are called *primitive concepts*. For our language, roles are always primitive since we do not have role definitions. The following is an example of a T-box in this formalism: Let Man, Human, Male and Mos (for "*m*an who has *o*nly *s*ons") be concept names and let child be a role name. The T-box consists of the following axioms:

$$\text{Man} = \text{Human} \sqcap \text{Male}$$
$$\text{Mos} = \text{Man} \sqcap \forall\text{child:Man}$$

That means that a man is human and male. A man who has only sons is a man such that all his children are male humans. Male and Human are primitive concepts while Man and Mos are defined concepts. Assume that we want to express a concept "*m*an who has *o*nly *m*ale *o*ff-springs", for short Momo. We can't just introduce a new role name off-spring because there would be no connection between the two primitive roles child and off-spring. But the intended meaning of off-spring is that it is the transitive closure of child. It seems quite natural to use a cyclic definition for Momo: A man who has only male off-springs is himself a man and all his children are men who have only male off-springs, i.e.,

$$\text{Momo} = \text{Man} \sqcap \forall\text{child:Momo}.$$

This is a very simple cyclic definition. In general, cycles in terminologies are defined as follows. Let A, B be concept names and let T be a T-box. We say that A *directly uses* B in T iff B appears on the right hand side of the definition of A. Let *uses* denote the transitive closure of the relation *directly uses*. Then T contains a *terminological cycle* iff there exists a concept name A in T such that A uses A.

The next definition gives a model-theoretic semantics for the language introduced in Definition 2.1.

Definition 2.2. (interpretations and models)
An *interpretation* I consists of a set dom(I), the domain of the interpretation, and an interpretation function which associates with each concept name A a subset A^I of dom(I), and with each role name R a binary relation R^I on dom(I), i.e., a subset of dom(I) × dom(I). The sets A^I, R^I are called extensions of A, R with respect to I.
The interpretation function – which gives an interpretation for atomic terms – can be extended to arbitrary terms as follows: Let C, D be concept terms and R be a role name. Assume that C^I and D^I are already defined. Then

$$(C \sqcap D)^I := C^I \cap D^I,$$
$$(\forall R{:}C)^I := \{\, x \in \text{dom}(I);\ \text{for all } y \text{ such that } (x,y) \in R^I \text{ we have } y \in C^I \,\}.$$

An interpretation I is a *model* of the T-box T iff it satisfies

$A^I = D^I$ for all terminological axioms A = D in T. ❑

The semantics we have just defined is not restricted to non-cyclic terminologies. But for cyclic terminologies this kind of semantics – which will be called "descriptive semantics" in the following – may seem unsatisfactory. One might think that the extension of a defined concept should be completely determined by the extensions of the primitive concepts and roles. This is the case for non-cyclic terminologies. More precisely, let T be a T-box

containing the primitive concepts $P_1, ..., P_n$ and the roles $R_1, ..., R_m$. If T doesn't contain cycles, then any interpretation $P_1^I, ..., P_n^I, R_1^I, ..., R_m^I$ of the primitive concepts and roles can uniquely be extended to a model of T (see e.g., (Nebel 1989a), Section 3.2.4). If T contains cycles, a given interpretation of all primitive concepts and roles[4] may have different extensions to models of T.

Example 2.3. Let R be a role name and B, P be concept names.[5] The terminology T consists of the single axiom $B = P \sqcap \forall R{:}B$.
We consider the following primitive interpretation: $dom(I) := \{ a, b, c, d \} =: P^I$ and $R^I := \{ (a,b), (c,d), (d,d) \}$. It is easy to see that this interpretation has two different extensions to models of T. The defined concept B may be interpreted as { a, b } or as { a, b, c, d }. Please note that individuals without R^I-successors are in the extension $(\forall R{:}C)^I$ of the term $\forall R{:}C$, no matter how C may be interpreted. ❑

The example also demonstrates that, with respect to descriptive semantics, the above construction $B = P \sqcap \forall R{:}B$ does not express the value-restriction $B = \forall R^*{:}P$ for the reflexive-transitive closure R^* of R. This implies that our definition of the concept Momo from above is not correct, if we use descriptive semantics.

For these reasons we shall now consider alternative types of semantics for terminological cycles.

3. Fixed-point Semantics for Terminological Cycles

A terminology may be considered as a parallel assignment where the defined concepts are the variables and the primitive concepts and roles are parameters.

Example 3.1. Let R, S be role names and A, B, P, Q be concept names, and let T be the terminology $A = Q \sqcap \forall S{:}B$, $B = P \sqcap \forall R{:}B$. We consider the following primitive interpretation I, which fixes the values of the parameters P, Q, R, S: $dom(I) := \{ a_0, a_1, a_2, ... \}$, $P^I := \{ a_1, a_2, a_3, ... \}$, $Q^I := \{ a_0 \}$, $R^I := \{ (a_{i+1},a_i); i \geq 1 \}$, and $S^I := \{ (a_0,a_i); i \geq 1 \}$.
For given values of the variables A, B, the parallel assignment $A := Q \sqcap \forall S{:}B$, $B := P \sqcap \forall R{:}B$ yields new values for A, B. If A and B are interpreted as the empty set, an application of the assignment T yields the values $\emptyset$ for A and $\{ a_1 \}$ for B. If we reapply the assignment to these values we obtain $\emptyset$ for A and $\{ a_1, a_2 \}$ for B. ❑

In the general case, a terminology T together with a primitive interpretation I defines a function T_I: $(2^{dom(I)})^n \rightarrow (2^{dom(I)})^n$, where $2^{dom(I)}$ denotes the set of all subsets of dom(I) and n is the number of defined concepts in T. For the above example we have seen that $T_I(\emptyset,\emptyset) = (\emptyset, \{ a_1 \})$ and $T_I(\emptyset,\{ a_1 \}) = (\emptyset,\{ a_1, a_2 \})$.

A primitive interpretation together with an element $\underline{A}$ of $(2^{dom(I)})^n$, which gives the extensions of the defined concepts, yields an interpretation of T. Obviously, this interpretation is a model of T if and only if $\underline{A}$ is a fixed-point of the function T_I, i.e., if and only if $T_I(\underline{A}) = \underline{A}$. In our example, the element $(\{ a_0 \},\{ a_1, a_2, a_3, ... \})$ of $(2^{dom(I)})^2$ is a fixed-point of T_I. If we extend I by defining $A^I := \{ a_0 \}$, $B^I := \{ a_1, a_2, a_3, ... \}$, we obtain a model of T.

One may now ask whether any primitive interpretation I can be extended to a model of T, or equivalently, whether any function T_I has a fixed-point. The answer is yes, because $(2^{dom(I)})^n$, ordered componentwise by inclusion, is a complete lattice (i.e., a partially ordered set where any subset has a least upper bound) and the functions T_I are monotonic (i.e., $\underline{A} \subseteq \underline{B}$ implies $T_I(\underline{A}) \subseteq T_I(\underline{B})$). More precisely, this implies that T_I has a least fixed-point $lfp(T_I)$, a greatest fixed-point $gfp(T_I)$, and possibly other fixed-points which lie between the least and the greatest fixed-point (see e.g., (Schmidt 1986), Chapter 6, and (Lloyd 1987), Chapter 1, §5).

Definition 3.2. Let T be a terminology, possibly containing terminological cycles.
(1) The *descriptive semantics* allows all models of T as admissible models.
(2) The *least fixed-point semantics (lfp-semantics)* allows only those models of T which come from the least fixed-point of a function T_I (lfp-models).
(3) The *greatest fixed-point semantics (gfp-semantics)* allows only those models of T which come from the greatest fixed-point of a function T_I (gfp-models). ❑

Any primitive interpretation I can uniquely be extended to a lfp-model (gfp-model) of T. In Example 2.3, the extension of I which interprets B as { a, b } is a lfp-model of T and the extension which interprets B as { a, b, c, d } is a gfp-model of T. It is easy to see that, for cycle-free terminologies, lfp-, gfp- and descriptive semantics coincide (see (Nebel 1989a), p.137,138).

The next question is how lfp-models (gfp-models) can be constructed from a given primitive interpretation. (Nebel 1987,1989,1989a) claimed that the functions T_I are even ω-continuous (i.e., for any chain $\underline{A}^{(0)} \subseteq \underline{A}^{(1)} \subseteq ...$, one has $\bigcup_{i\geq 0} T_I(\underline{A}^{(i)}) = T_I(\bigcup_{i\geq 0}\underline{A}^{(i)})$), and that thus $lfp(T_I) = \bigcup_{i\geq 0} T_I^i(bottom)$, where bottom denotes the least element of $(2^{dom(I)})^n$, namely the n-tuple $(\emptyset,...,\emptyset)$. Unfortunately, this is not true.

Proposition 3.3. In general, we may have $lfp(T_I) \neq \bigcup_{i\geq 0} T_I^i(bottom)$.
PROOF. We consider Example 3.1. It is easy to see that $T_I^i(\emptyset,\emptyset) = (\emptyset,\{ a_1, a_2, ..., a_i \})$. Thus $\bigcup_{i\geq 0} T_I^i(\emptyset,\emptyset) = (\emptyset,\{ a_i; i \geq 1 \})$ which is not a fixed-point since $T_I(\emptyset,\{ a_i; i \geq 1 \}) = (\{ a_0 \},\{ a_i; i \geq 1 \})$. ❑

In this example, the least fixed-point is reached by

4. In the following such a partial interpretation will be called primitive interpretation.
5. We shall no longer use intuitive names for concepts and roles since I agree with (Brachman & Schmolze 1985), p.176, that "suggestive names can do more harm than good in semantic networks and other representation schemes." Suggestive names may seemingly exclude models which are admissible with respect to the formal semantics.

applying T_I once more after building the limit. In general, one may need several limit processes to obtain the least fixed-point (see (Lloyd 1987), p.29). On the other hand, I was able to show that the greatest fixed-point can always be reached by a single limit process.

Proposition 3.4. The functions T_I are always downward ω-continuous, i.e., for any descending chain $\underline{A}^{(0)} \supseteq \underline{A}^{(1)} \supseteq \ldots$ we have $\bigcap_{i\geq 0} T_I(\underline{A}^{(i)}) = T_I(\bigcap_{i\geq 0}\underline{A}^{(i)})$. Consequently, the greatest fixed-point may be obtained as $gfp(T_I) = \bigcap_{i\geq 0} T_I^i(top)$, where top := (dom(I),...,dom(I)).
PROOF. See (Baader 1990). ❑

The two propositions show that, from a constructive point of view, the gfp-semantics should be preferred. However, if dom(I) is finite, the greatest and the least fixed-point can be reached after a finite number of applications of T_I.

An important service terminological representation systems provide is computing the subsumption hierarchy.

Definition 3.5. (subsumption of concepts)
Let T be a terminology and let A, B be concept names.

$$A \sqsubseteq_T B \text{ iff } A^I \subseteq B^I \text{ for all models I of T,}$$
$$A \sqsubseteq_{lfp,T} B \text{ iff } A^I \subseteq B^I \text{ for all lfp-models I of T,}$$
$$A \sqsubseteq_{gfp,T} B \text{ iff } A^I \subseteq B^I \text{ for all gfp-models I of T.}$$

In this case we say that B *subsumes* A in T w.r.t. descriptive semantics (resp. lfp-semantics, gfp-semantics). ❑

4. Characterization of the Semantics Using Finite Automata

Before we can associate a finite automaton $\mathcal{A}_T$ to a terminology T we must transform T into some kind of normal form. It is easy to see that the concept terms $\forall R{:}(B \sqcap C)$ and $(\forall R{:}B) \sqcap (\forall R{:}C)$ are equivalent. Hence any concept term can be transformed into a finite conjunction of terms of the form $\forall R_1{:}\forall R_2{:}\ldots\forall R_n{:}A$, where A is a concept name. We shall abbreviate the prefix "$\forall R_1{:}\forall R_2{:}\ldots\forall R_n$" by "$\forall W$" where $W = R_1R_2\ldots R_n$ is a word over $\mathbf{R}_T$, the set of role names occurring in T. In the case n = 0 we also write "$\forall\varepsilon{:}A$"[6] instead of simply "A". For an interpretation I and a word $W = R_1R_2\ldots R_n$, let W^I denote the composition $R_1^I \circ R_2^I \circ \ldots \circ R_n^I$ of the binary relations R_1^I, R_2^I, ..., R_n^I. The term ε^I denotes the identity relation, i.e., $\varepsilon^I = \{ (d,d); d \in dom(I) \}$.

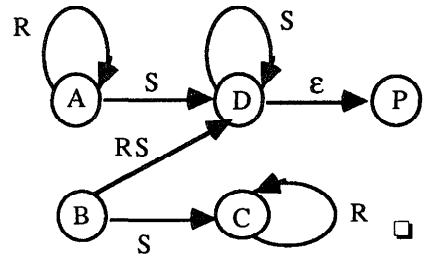
Let T be a terminology where all terms are normalized as described above.

Definition 4.1. The generalized (nondeterministic) automaton $\mathcal{A}_T$ is defined as follows: The alphabet of $\mathcal{A}_T$ is the set $\mathbf{R}_T$ of all role names occurring in T; the states of $\mathcal{A}_T$ are the concept names occurring in T; a terminological axiom of the form $A = \forall W_1{:}A_1 \sqcap \ldots \sqcap \forall W_k{:}A_k$ gives rise to k transitions, where the transition from A to A_i is labeled by the word W_i. ❑

The automaton $\mathcal{A}_T$ is called "generalized" because transitions are labeled by words over the alphabet and not only by symbols of the alphabet. However, it is well-known that any generalized finite automaton can be transformed into an equivalent finite automaton (see (Manna 1974), p. 9). Definition 4.1 will now be illustrated by an example.

Example 4.2. (A normalized terminology and the corresponding automaton)

$A = \forall R{:}A \sqcap \forall S{:}D$ $\quad$ $B = \forall RS{:}D \sqcap \forall S{:}C$
$C = \forall R{:}C$ $\quad$ $D = \forall S{:}D \sqcap \forall\varepsilon{:}P$



❑

A finite path (resp. infinite path) in a generalized automaton will be described by the initial state of the path and the finite sequence of transition labels followed by the terminal state of the path (resp. by the initial state and the infinite sequence of transition labels). For example, A; R, R, S, S, S, ε; P is a finite path in the above automaton and B; RS, S, S, S, ... is an infinite path. The label of the finite path is the finite word RRSSS while the label of the infinite path is the infinite word RSSSS... . For two states p, q of an automaton let L(p,q) denote the set of all finite words which are labels of paths fom p to q and let U(p) be the set of all finite[7] and infinite words which are labels of infinite paths with initial state p. In the example, $L(A,P) = R^*SS^* = \{ R^nSS^m; n,m \geq 0 \}$ and $U(A) = \{ RRR\ldots \} \cup \{ R^nSSS\ldots; n \geq 0 \}$. We are now ready to characterize the gfp-semantics.

Theorem 4.3. Let T be a terminology and let $\mathcal{A}_T$ be the corresponding automaton. Let I be a gfp-model of T and let A, B be concept names occurring in T.
(1) For any $d \in dom(I)$ we have $d \in A^I$ iff *for all primitive concepts P, all words $W \in L(A,P)$, and all individuals $e \in dom(I)$, $(d,e) \in W^I$ implies $e \in P^I$.*
(2) Subsumption in T can be reduced to inclusion of regular languages defined by $\mathcal{A}_T$. More precisely, $A \sqsubseteq_{gfp,T} B$ iff *$L(B,P) \subseteq L(A,P)$ for all primitive concepts P.*
PROOF. The "if" direction of (1) is proved by induction on the positive integer n such that d is not in the A-component of $T_I^n(top)$. The "only if" direction is proved by induction on the length of a path from A to P with label W. (2) is a relatively easy consequence of (1). See (Baader 1990) for a complete proof. ❑

The theorem can intuitively be understood as follows: The language L(A,P) stands for the possibly infinite number of constraints of the form ∀W:P which the terminology imposes on A. The more constraints are imposed the smaller the concept is. In the example, B

6. "ε" denotes the empty word.

7. An infinite path of the form p; W_1, ..., W_n, ε, ε, ε, ... has the finite word $W_1\ldots W_n$ as label. This word is also an element of U(p).

subsumes A w.r.t. gfp-semantics since $L(B,P) = RSS^*$ is a subset of $L(A,P) = R^*SS^*$. For the terminology $B = P \sqcap \forall R{:}B$ of Example 2.3, $L(B,P) = R^* = \{ R^n; n \geq 0 \}$. Hence it is an immediate consequence of part (1) of the theorem that this terminology – if interpreted with gfp-semantics – expresses value-restriction with respect to the reflexive-transitive closure of R.

Corollary 4.4. The problem of determining subsumption w.r.t. gfp-semantics is PSPACE-complete.
PROOF. We have already seen that subsumption can be reduced to inclusion of regular languages. On the other hand, it is relatively easy to show that inclusion of regular languages can be reduced to subsumption determination w.r.t. gfp-semantics. It is well-known (see (Garey & Johnson 1979)) that inclusion of regular languages is PSPACE-complete.[8] ❑

This shows that, even for our very small language, subsumption determination w.r.t. gfp-semantics is rather hard from a computational point of view. On the other hand, (Nebel 1989b) has shown that, even without cycles, this language has a co-NP-complete subsumption problem.

We shall now consider the lfp-semantics.

Proposition 4.5. Let T be a terminology and let $\mathcal{A}_T$ be the corresponding automaton. Let I be an lfp-model of T and let A be a concept name occurring in T. For any $d_0 \in dom(I)$ we have $d_0 \in A^I$ iff *the following two properties hold:*
(P1) *For all primitive concepts P, all words $W \in L(A,P)$, and all individuals $e \in dom(I)$, $(d_0,e) \in W^I$ implies $e \in P^I$.*
(P2) *For all infinite paths A; $W_1, W_2, W_3, \ldots$ and all individuals $d_1, d_2, d_3, \ldots$ there exists $n \geq 1$ such that $(d_{n-1},d_n) \notin W_n^I$.*
PROOF. The proof is more difficult than the proof of part (1) of Theorem 4.3 since it requires transfinite induction (see (Baader 1990)). ❑

As a consequence of P2 of the proposition, ε-cycles in $\mathcal{A}_T$ – i.e., non-empty paths of the form B; ε, ..., ε; B – are important for the lfp-semantics. In particular, it is easy to show that the concept A is *inconsistent w.r.t. lfp-semantics* – i.e., it has the empty extension in all lfp-models of T – if and only if there exists a path A; ε, ..., ε; B such that B is the initial state of an ε-cycle. Because of this phenomenon, the automaton $\mathcal{A}_T$ has to be modified before we can express subsumption w.r.t. lfp-semantics.

We add a new state Q_{loop} to $\mathcal{A}_T$, a transition with label ε from Q_{loop} to Q_{loop}, and for each role R in T a transition with label R from Q_{loop} to Q_{loop}. For any state B of $\mathcal{A}_T$ lying on an ε-cycle we add a transition with label ε from B to Q_{loop}, and for any primitive concept P we add a transition with label ε from Q_{loop} to P. This modified automaton will be called $\mathcal{B}_T$.

8. This is only true if the languages are given by nondeterministic automata. With respect to the size of deterministic automata the problem could be solved in quadratic time.

Theorem 4.6. Let T be a terminology and let $\mathcal{B}_T$ be the corresponding modified automaton. Then $A \sqsubseteq_{lfp,T} B$ iff *$U(B) \subseteq U(A)$ and $L(B,P) \subseteq L(A,P)$ for all primitive concepts P.*
PROOF. See (Baader 1990). ❑

In Example 4.2, B does not subsume A w.r.t. lfp-semantics since U(B) contains the infinite word SRRR... which is not in U(A). It is not hard to show that the inclusion problem "$U(B) \subseteq U(A)$", which refers to languages of finite and infinite words, can be reduced to an ordinary inclusion problem for regular languages. On the other hand, inclusion of regular languages can also be reduced to subsumption w.r.t. lfp-semantics (see (Baader 1990)).

Corollary 4.7. The problem of determining subsumption w.r.t. lfp-semantics is PSPACE-complete. ❑

For the descriptive semantics, the characterization of subsumption as well as the proof of its correctness is more involved. Infinite paths are still important but it is not enough to consider just their labels. The states which are reached infinitely often by the path are also significant. An infinite path which has initial state A and reaches the state C infinitely often will be represented in the form A, U_0, C, U_1, C, U_2, C, ..., where the U_i are labels of non-empty paths[9] from A to C for i = 0 and from C to C for i > 0.

Theorem 4.8. Let T be a terminology and let $\mathcal{A}_T$ be the corresponding automaton. Then $A \sqsubseteq_T B$ iff *the following two properties hold:*
(P1) *For all primitive concepts P, $L(B,P) \subseteq L(A,P)$.*
(P2) *For all defined concepts C and all infinite paths of the form B, U_0, C, U_1, C, U_2, C, ... there exists $k \geq 0$ such that $U_0U_1...U_k$ is the label of a path from A to C.*
PROOF. See (Baader 1990). ❑

It is not at all obvious how to decide P2 for given states A, B, C of a generalized nondeterministic automaton. Fortunately, this problem can be reduced (see (Baader 1990)) to an inclusion problem for a certain class of languages of infinite words, and this class has already been considered in the context of monadic second-order logic (see (Büchi 1960)). A solution to the inclusion problem for this class is also not obvious, but there is a theorem due to Büchi and McNaughton (see (Eilenberg 1974), p.382[10]) which implies that the class is closed under intersection and complement. But then the inclusion problem can be reduced to the emptiness problem for these languages as follows: $L_1 \subseteq L_2$ iff $\overline{L_2} \cap L_1 = \emptyset$. Finally, it is rather easy to solve the emptiness problem. More precisely, it follows from a result in (Sistla, Vardi & Wolper 1987) that the inclusion problem for languages accepted by Büchi automata is PSPACE-complete. This shows that the problem of subsumption determination

9. Non-empty means that the path uses at least one transition. Nevertheless, U_i can be empty if all transition of the path are labeled with ε.
10. The proof is constructive; but it takes eight pages, which shows that we are dealing with a hard problem.

w.r.t. descriptive semantics is decidable with polynomial space (see also (Nebel 1990)).

5. Conclusion

We have considered a small terminological language, because for this language the meaning of terminological cycles with respect to different kinds of semantics and, in particular, the important subsumption relation could be characterized with the help of finite automata. These results may help to decide what kind of semantics is most appropriate for cyclic definitions, not only for this small language, but also for suitably extended languages. As it stands, the gfp-semantics comes off best. The characterization given in Theorem 4.3 is easy and has an obvious intuitive interpretation. Furthermore, important constructs – such as value-restriction with respect to the reflexive-transitive closure of a role – can easily be expressed. The lfp-semantics is less constructive and the modifications of the automaton which are necessary to characterize subsumption are not obvious. For the descriptive semantics one has to consider certain languages of infinite words which are more difficult and less intuitive than the regular languages which occur in the context of gfp-semantics.

This research can be continued in two directions. Firstly, one may try to extend the results to cyclic definitions in larger languages. As a first step in this direction, the results for gfp-semantics were extended to cycles in the language $\mathcal{FL}^-$ of (Levesque-Brachman 1987). Hybrid inferences such as realization can also be handled in this context (see (Baader 1990)). Secondly, one can use a larger language, but restrict cycles to the small language. One idea in this direction is to extend a given language by value-restrictions of the form $\forall L{:}P$ where L is a regular language over the alphabet of role names. In accordance with part (1) of Theorem 4.3, the semantics of this construct should be defined as $(\forall L{:}P)^I := \{ d \in dom(I);$ for all words $W \in L$ and all individuals $e \in dom(I)$, $(d,e) \in W^I$ implies $e \in P^I \}$. For example, $\forall RR^*{:}P$ would express value-restriction with respect to the transitive closure of the role R (RR^* is the regular language $\{ R^n; n \geq 1 \}$).

6. References

Aho, A.V., and Ullman, J.D. 1979. Universality of Data Retrieval Languages. In Proceedings of the 6th ACM Symposium on Principles of Programming Languages, 110–120.

Baader, F. 1990. Terminological Cycles in KL-ONE-based KR-languages. Research Report, RR-90-01, DFKI, Kaiserslautern.

Brachman, R.J., and Schmolze, J.G. 1985. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science* 16: 171–216.

Brachman, R.J., Pigman-Gilbert, V., and Levesque, H.J. 1985. An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts in KRYPTON. In Proceedings of the 9th International Joint Conference on Artificial Intelligence, 532–539, Los Angeles.

Büchi, J.R. 1960. On a Decision Method in Restricted Second Order Arithmetic. In Proceedings of the 1960 Congress on Logic, Methodology and Philosophy of Science, 1–11, Stanford.

Eilenberg, S. 1974. *Automata, Languages and Machines, Vol. A.* New York/London: Academic Press.

Garey, M.R., and Johnson, D.S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco: Freeman.

Gurevich, Y., and Shelah, S. 1986. Fixed-point Extensions of First-Order Logic. *Annals of Pure and Applied Logic* 32: 265–280.

Immerman, N. 1982. Relational Queries Computable in Polynomial Time. In Proceedings of the 4th ACM Symposium on the Theory of Computing, 147–152.

Kaczmarek, T.S., Bates, R., and Robins, G. 1986. Recent Developments in NIKL. In Proceedings of the 5th National Conference of the American Association for Artificial Intelligence, 978–987, Philadelphia.

Levesque, H.J., and Brachman, R.J. 1987. Expressiveness and Tractability in Knowledge Representation and Reasoning. *Computational Intelligence* 3: 78–93.

Lloyd, J.W. 1987. *Logic Programming, Second, Extended Edition.* Berlin: Springer Verlag.

McNaughton, R. 1966. Testing and Generating Infinite Sequences by a Finite Automaton. *Information and Control* 9: 521–530.

MacGregor, R., and Bates, R. 1987. The Loom Knowledge Representation Language. Technical Report ISI/RS-87-188, Information Science Institute, Univ. of Southern California.

Manna, Z. 1974. *Mathematical Theory of Computation.* New York: McGraw-Hill.

Nebel, B. 1987. On Terminological Cycles. KIT Report 58, Technische Universität Berlin.

Nebel, B. 1989. On Terminological Cycles. In Preprints of the Workshop on Formal Aspects of Semantic Networks, Two Harbors.

Nebel, B. 1989a. Reasoning and Revision in Hybrid Representation Systems. PhD diss, Universität des Saarlandes, Saarbrücken.

Nebel, B. 1989b. Terminological Reasoning is Inherently Intractable. IWBS Report 82, IBM Deutschland.

Nebel, B. 1990. Terminological Cycles: Semantics and Computational Properties. To appear in Sowa, J. ed. 1990. *Formal Aspects of Semantic Networks.*

Schmidt, D.A. 1986. *Denotational Semantics: A Methodology for Language Development.* Boston: Allyn and Bacon.

Sistla, A.P., Vardi, M.Y., and Wolper, P. 1987. The Complementation Problem for Büchi Automata with Applications to Temporal Logic. *Theoretical Computer Science* 49: 217–237.

Vardi, M. 1982. Complexity of Relational Query Languages. In Proceedings of the 4th ACM Symposium on the Theory of Computing, 137–146.

Vielle, L. 1989. Recursive Query Processing: The Power of Logic. *Theoretical Computer Science* 69: 1–53.