Tractable Reasoning in a Universal Description Logic: Extended Abstract*

Klaus Schild
German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, FRG
e-mail: schild@dfki.uni-sb.de

1 Introduction

Description logics (also called *terminological logics* or *concept languages*) have been designed for the logical reconstruction and specification of knowledge representation systems descending from KL-ONE such as BACK, CLASSIC, $\mathcal{KRIS}$, and LOOM.[1] These systems are used to make the terminology of an application domain explicit and then to classify these definitions automatically into a taxonomy according to semantic relations like subsumption and equivalence. More precisely, automatic classification refers to the ability to insert a new concept into the taxonomy in such a way that it is directly linked to the most specific concept it is subsumed by and to the most general concept it in turn subsumes. Terminological knowledge representation systems thereby support the task to formalize an application in at least two respects. On the one hand, they urge the user to isolate the intrinsic concepts of the application; on the other hand they may detect hidden subsumption and equivalence relations between definitions or may even detect that a definition is incoherent.

A model of the application is then given by associating special objects of the domain with the concepts of the terminology. The systems mentioned above in turn automatically classify these objects with respect to the given terminology and to those membership relations which have been asserted explicitly. In this case, however, automatic classification refers to the ability to find the most specific concept the object is a member of.

Terminologies comprise two different kinds of terms, viz. so-called *concepts* and *roles*. The former are intended to represent classes of objects of a given domain, while the latter represent binary relations over this domain. Concepts can either be simple *concept names*, representing not further specified classes of objects, or structured by means of a fixed set of *concept structuring primitives*. Common concept structuring primitives are *concept conjunction* $\sqcap$ and universal quantification $\forall R{:}C$ over a role R. Concept conjunction is to be interpreted as set intersection, while the concept $\forall R{:}C$ denotes all those objects d of the domain for which each object related to d by the role R is a member of the concept C. Although there exist many other concept structuring primitives, it is commonly accepted that these two should be part of each concept language. In contrast to concepts, roles are often taken to be atomic, i.e., there are no roles other than *role names*. The standard concept language $\mathcal{ALC}$, for instance, does not comprise any role structuring primitives. However, in addition to those mentioned above, this language comprises *concept disjunction* $\sqcup$, *concept negation* $\neg$ as well as existential quantification $\exists R{:}C$ over a role R as concept structuring primitives. For details the reader is referred to [Schmidt-Schauß and Smolka, 1991].

Definitions are given by associating a concept or role T with a concept name (resp., role name) TN. Such a definition is represented by the expression $TN \doteq T$ and is called *concept* and *role introduction* respectively. *Terminologies* are just finite sets of concept and role introductions such that each concept and role name is defined at most once, i.e., for every concept and role name TN there exists at most one concept or role introduction the left-hand side of which is TN.

As already mentioned, a model of application domain is described in terms of the given terminology. More precisely, specific objects of the domain and pairs of objects can be associated with concepts and roles of the terminology, where these objects are syntactically represented by so-called *individual names*. It can either be asserted that an individual name a is an instance of a concept C or that it is related to another individual name, say, b, by a role R. Such assertions are called *assertional axioms* and are represented by the expressions $a{:}C$ and $(a,b){:}R$ respectively. A finite set of assertional axioms forms a *knowledge base*.

From a theoretical point of view, the computational service provided by terminological knowledge representation systems can be reduced to answer queries of the following form with respect to a knowledge base $\mathcal{KB}$ and to a terminology $\mathcal{T}$: a *query* can be an assertional axiom or an *inclusion axiom* of the form $T_1 \sqsubseteq T_2$, where T_1 and T_2 are either two concepts or two roles. The meaning of such a query Q posed with respect to $\mathcal{KB}$ and $\mathcal{T}$ is usually given in terms of so-called interpretations and models. An *interpretation* $\mathcal{I}$ consists of a *domain* $\Delta^{\mathcal{I}}$ and a *val-*

*This work was supported by a grant from the *Deutsche Forschungsgemeinschaft* (DFG).

[1]For a good overview of the so-called KL-ONE family the reader is referred to [Woods and Schmolze, 1992]; for KL-ONE itself cf. [Brachman and Schmolze, 1985].

uation $\mathcal{V}$ over $\Delta^{\mathcal{I}}$ along with an *interpretation function* $.^{\mathcal{I}}$. The valuation $\mathcal{V}$ over $\Delta^{\mathcal{I}}$ maps each concept name to a subset of $\Delta^{\mathcal{I}}$ and each role name to a binary relation over $\Delta^{\mathcal{I}}$. Individual names, however, are mapped to singleton sets containing exactly one element of $\Delta^{\mathcal{I}}$. The interpretation function $.^{\mathcal{I}}$, on the other hand, just extends $\mathcal{V}$ to deal with arbitrary concepts and roles in such a way that all concept and role structuring primitives are interpreted properly. The concept structuring primitives $\sqcap$, $\sqcup$, $\neg$, for instance, are to be interpreted as the corresponding set operations on $\Delta^{\mathcal{I}}$, while the interpretation of the concept $\forall R{:}C$ is defined inductively as follows: if $C^{\mathcal{I}}$ and $R^{\mathcal{I}}$ have already been defined, then $(\forall R{:}C)^{\mathcal{I}}$ is $\{d \in \Delta^{\mathcal{I}} : \forall e(\langle d,e\rangle \in R^{\mathcal{I}}), e \in C^{\mathcal{I}}\}$.

An interpretation $\mathcal{I}$ is then said to be a *model* of the inclusion axiom $T_1 \sqsubseteq T_2$ just in case that $T_1^{\mathcal{I}} \subseteq T_2^{\mathcal{I}}$ and, if a and b are individual names such that $a^{\mathcal{I}}$ is $\{\underline{a}\}$ and $b^{\mathcal{I}}$ is $\{\underline{b}\}$, then $\mathcal{I}$ is a model of the assertional axiom $a{:}C$ (resp., of $(a,b){:}R$) just in case that $\underline{a} \in C^{\mathcal{I}}$ (resp., $\langle \underline{a}, \underline{b}\rangle \in R^{\mathcal{I}}$). Not very surprising, an interpretation is a *model* of $\mathcal{KB}$ and $\mathcal{T}$ if it is a model of each of the elements of $\mathcal{KB}$ and $\mathcal{T}$. Now, Q is said to be *entailed* by $\mathcal{KB}$ and $\mathcal{T}$, written $\mathcal{KB} \models_{\mathcal{T}} Q$, if and only if every interpretation which is a model of $\mathcal{KB}$ and $\mathcal{T}$ is a model of Q as well. Moreover, we say that T_2 *subsumes* T_1 with respect to $\mathcal{T}$ if and only if it holds that $\emptyset \models_{\mathcal{T}} T_1 \sqsubseteq T_2$.

2 Terminological Reasoning is Inherently Intractable

Unfortunately, answering such queries is in most cases provably intractable, at least in terms of computational worst case complexity. This applies, for instance, to the basic inference of KL-ONE, although originally claimed to be computationally tractable. In fact, Schmidt-Schauß [1989] proved that there exists no algorithm at all which decides whether one concept of KL-ONE subsumes another one or not, even with respect to empty terminologies.

Moreover, in [Schild, 1993, 94a], , it is proved that in case of the standard concept language $\mathcal{ALC}$, every algorithm capable of deciding whether one concept subsumes another one or not uses more than polynomial time in the worst case if at least one (possibly recursive) concept introduction is taken into account. Notably, this result holds no matter which of the usual kinds of semantics for recursive concept introductions is presupposed, viz. either *descriptive semantics* or *least* or *greatest fixed point semantics*, as Nebel [1991] called them.

It is also known that even in case of the minimal concept language (comprising no concept and role structuring primitives other than concept conjunction and universal quantification over role names), there exists no polynomial time algorithm which decides with respect to acyclic terminologies whether one concepts subsumes another one or not, unless P = NP [Nebel, 1990].

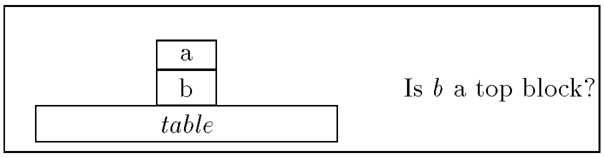

Figure 1: A sample blocks world.

$$\left\{\begin{array}{l} \forall x.block(x) \Leftrightarrow x = a \vee x = b, \\ a \neq b, a \neq table, b \neq table, \\ \forall x \forall y.on(x,y) \;\Leftrightarrow\; (x = a \wedge y = b) \\ \qquad\qquad\qquad\quad \vee \;\; (x = b \wedge y = table) \end{array}\right\}$$
$$\overset{?}{\models} \; block(b) \wedge \neg\exists x.block(x) \wedge on(x,b)$$

Figure 2: Representing the sample blocks world by first-order formulae.

3 Model Checking Versus Theorem Proving

In the previous section, we have seen that, as Woods and Schmolze [1992] put it, "the surfeit of intractability results seems to have reached its logical end with the conclusion that practically everything of any use is intractable (in the worst case)." Recently, Halpern and Vardi [1991] proposed a possible solution to this very problem of knowledge representation. As a starting point, they re-examined the traditional approach to knowledge representation, going back to McCarthy [1968]. According to this approach the world to be modeled should be represented by a finite set of formulae of some given logic, preferably first-order logic. If a question to be answered is then formulated within the same logic, the answer depends on whether this formula is a *logical consequence* of the collection of formulae representing the world or not. In other words, it is checked whether *every* semantic structure which is a model of each of the formulae representing the world is also a model of formula corresponding to the question.

We shall illustrate this traditional approach to knowledge representation by means of an example, drawn from the famous blocks world. Suppose, for instance, we would like to represent a blocks world involving two blocks, say, a and b, where a lies on b and the latter in turn lies on a table. Suppose, furthermore, we would like to know whether b is a top block or not. Figure 1 depicts exactly this situation, while Figure 2 gives its representation in terms of first-order logic in the traditional way just described.

McCarthy's approach, however, gives rise to the problem that the need to represent all facts about the world in terms of some logic necessitates the use of very expressive logics such as full first-order logic. This, in fact, gives rise to difficulties because it is known that there exists no algorithm at all which generally decides logical consequence in full first-order logic [Church, 1936], and this remains true even when only finite interpretation domains are taken into consideration [Trahtenbrot, 1963].

At this very point Halpern and Vardi stressed that

$$\begin{array}{lcl} \mathcal{D}om & = & \{a, b, table\} \\ [\![block]\!] & = & \{a, b\} \\ [\![on]\!] & = & \{\langle a, b\rangle, \langle b, table\rangle\} \end{array}$$
$$\overset{?}{\models} \; block(b) \wedge \neg\exists x.block(x) \wedge on(x, b)$$

Figure 3: Representing the sample blocks world by a semantic structure.

in many cases the natural representation of a world to be modeled is a *semantic structure* rather than a collection of formulae. If, as in the traditional approach, queries are represented by formulae of a given logic, a query can be answered in this case depending on whether the formula representing the query is true in the *given* semantic structure or not. That is to say, it is checked whether the semantic structure is a model of the formula corresponding to the query. The fact that a (closed) formula α is true in a semantic structure $\mathcal{M}$ is usually indicated by $\mathcal{M} \models \alpha$. Resorting to this convention, Figure 3 gives such an alternative representation of the blocks world considered above.

In many cases this model checking approach has tremendous benefits, at least in terms of computational complexity. For instance, checking the truth of an arbitrary closed first-order formula[2] α in a finite semantic structure fixing the interpretation of all predicates and constants occurring in α is known to be *decidable* using at most polynomial space [Chandra and Merlin, 1977]. Recall that in contrast to this, there exists no algorithm at all which is able to decide whether an arbitrary formula of this kind is a logical consequence of a finite set of first-order formulae, even with only finite interpretation domains taken into account. However, it is also known that first-order model checking is still at least as hard as *any other* problem solvable using at most polynomial space, hence this problem is still very hard [Chandra and Merlin, 1977]. Anyway, Halpern and Vardi's intention was to forge a new approach to knowledge representation rather than to give concrete instances which allow for tractable inferences.

4 The Model Checking Approach to Terminological Reasoning

It should be clear that terminological knowledge representation, as described in the introduction, is committed to the traditional approach to knowledge representation rather than to the model checking approach. In [Schild, 1994b] we investigated the consequences of adapting Halpern and Vardi's model checking approach to terminological reasoning. It turned out that even in case of the most powerful description logic considered in the literature, answering queries become tractable just by replacing the usual kind of knowledge bases with single finite semantic structures fixing the interpretation of all *primitive* concepts and roles (i.e., those concept and role names which are mentioned somewhere in the terminology or in the query, but which are not defined).

[2]This formula should involve no function symbols other than constants.

$$\left\{\begin{array}{l} a{:}Block, b{:}Block, table{:}\neg Block, \\ (a, b){:}on, (b, table){:}on, \\ a{:}(\neg\exists on^{-1}{:}Block), table{:}(\neg\exists on{:}Block) \end{array}\right\}$$
$$\mathcal{T} = \{TopBlock \doteq Block \sqcap \neg\exists on^{-1}{:}Block\}$$
$$\overset{?}{\models}_{\mathcal{T}} \; b{:}TopBlock$$

Figure 4: Representing the sample blocks world by an $\mathcal{ALC}^{-1}$-KB.

$$\begin{array}{lcl} \mathcal{D}om & = & \{a, b, table\} \\ [\![Block]\!] & = & \{a, b\} \\ [\![on]\!] & = & \{\langle a, b\rangle, \langle b, table\rangle\} \end{array}$$
$$\mathcal{T} = \{TopBlock \doteq Block \sqcap \neg\exists on^{-1}{:}Block\}$$
$$\overset{?}{\models}_{\mathcal{T}} \; b{:}TopBlock$$

Figure 5: Representing the sample blocks world by a physical $\mathcal{ALC}^{-1}$-KB.

But before engaging into details, have a look at Figure 4, which shows how to represent the already familiar blocks world in terms of $\mathcal{ALC}$ together with the inverse of roles $^{-1}$, as it would be done traditionally. Observe, however, that this representation is *incomplete* in that it solely states that block a lies on block b, while the latter in turn lies on the table, but it is left open whether there is any other block lying on b or on the table. As a matter of fact, there is no way at all to give an *accurate* representation of our blocks world in terms of $\mathcal{ALC}$, even when augmented by the inverse of roles. This means, in this case the so-called *open world assumption*,[3] traditionally made for terminological reasoning, is a nuisance rather than an advantage.

Figure 5 modifies the just considered representation in the spirit of the model checking approach. A finite semantic structure is shown there which fixes the interpretation of each primitive concept and role of $\mathcal{T}$, that is, it fixes the interpretation of *Block* and *on*. Such a semantic structure is obviously nothing but a valuation along with a domain. When taken together with a domain, the syntactic representation of such a valuation is called *physical knowledge base*, emphasizing the fact that they are intended to replace customary knowledge bases. Now, suppose $\mathcal{V}$ is such a physical knowledge base with domain $\mathcal{D}om$, $\mathcal{T}$ is an arbitrary terminology, and Q is a query. Then $\mathcal{V} \models_{\mathcal{T}} Q$ is intended to mean that every interpretation extending $\mathcal{V}$ which is a model of $\mathcal{T}$ is a model of Q as well, where an interpretation $\mathcal{I}$ is said to *extend* a physical knowledge base $\mathcal{V}$ with domain $\mathcal{D}om$ just in case that $\Delta^{\mathcal{I}} = \mathcal{D}om$ and, moreover, $\cdot^{\mathcal{I}}$ interprets all those concept and role names handled

[3]In contrast to the *closed world assumption*, usually made for databases, the open world assumption does *not* assume that all those facts that are not explicitly mentioned (or that cannot be inferred) are taken to be false.

by $\mathcal{V}$ in exactly the same way as $\mathcal{V}$ does.

In [Schild, 1994b] we investigated the computational complexity of answering such queries with respect to physical knowledge bases in the description logic $\mathcal{U}$, introduced by Patel-Schneider [1987] as a universal description logic. This concept language is *universal* in the sense that it encompasses all others considered in the literature, except for those which comprise nonstandard facilities like defaults, for instance. In addition to those of $\mathcal{ALC}$, this language comprises *number restrictions* of the form $\exists^{\geq n} R{:}C$ and $\exists^{\leq m} R{:}C$ as well as *role value maps* of the form $R \leq S$ as concept structuring primitives. Number restrictions restrict the number of role fillers (i.e., those objects which are related to an object by a role), while role value maps impose restrictions on the fillers of two roles. The concept $R \leq S$ states that all fillers of the role R are also fillers of the role S. In addition, $\mathcal{U}$ admits of individual names to occurring in concepts. The role structuring primitives of $\mathcal{U}$ are the *identity role* ϵ, Boolean operations $\sqcap$, $\sqcup$, $\neg$ on roles, the inverse R^{-1} of a role, the composition $R \circ S$ of two roles, as well as the *transitive closure* R^{+} and the *reflexive-transitive closure* R^{*} of a role. For details cf. [Schild, 1994b] or [Patel-Schneider, 1987]. Notably, it is known that there cannot exist any algorithm which is capable of deciding subsumption between two concepts (or two roles) of $\mathcal{U}$, even with respect to empty terminologies [Schild, 1988].

The main result of [Schild, 1994b] is that even in this language $\mathcal{V} \models_{\mathcal{T}} Q$ can be decided in polynomial time provided that each of the following conditions is satisfied:

(a) $\mathcal{V}$ has a finite domain and specifies all concept and role names occurring in $\mathcal{T}$ and Q except for those which are defined in $\mathcal{T}$;

(b) Roles are not defined recursively;

(c) Concepts can be defined recursively, but then they must occur in their definition[4] *positively*, i.e., they must occur in the scope of an even number of negations, where $\exists^{\leq m} R{:}$ counts also as a negation. Moreover, each recursive definition must be given either least or greatest fixed point semantics, not necessarily in a uniform way.

Of course, each of these conditions calls for some comment. Condition (b) is commonly presupposed for terminological reasoning, while condition (c) constitutes the most liberal restriction on recursive concept definitions considered in the literature. The most important condition, however, is the first one in that it ensures all primitive concepts and roles to be specified extensionally. This restriction does make sense as these concepts and roles are exactly those which are not further specified according to the semantics. It can easily be verified that the sample query of Figure 5 obeys each of the three conditions above.

The employed algorithm capable of deciding $\mathcal{V} \models_{\mathcal{T}} Q$ in polynomial time just mimics the semantics of the concept and role structuring primitives of $\mathcal{U}$, storing already evaluated ones. To deal with recursive concept definitions, however, we exploited a technique for computing least and greatest fixed points due to Emerson and Lei [1986].

It turned out that even when relaxing condition (a) in such a way that $\mathcal{V}$ is solely required to have a finite domain, $\mathcal{V} \models_{\mathcal{T}} Q$ is still decidable in the universal description logic $\mathcal{U}$. In fact, we proved that in this case the computational complexity is essentially the same as the one of deciding ordinary subsumption between two concepts with respect to acyclic terminologies in the *minimal* concept language.[5]

We also investigated the consequences of incorporating some limited kind of incomplete knowledge by means of Reiter's *null values* [Reiter, 1984]. It turned out that, when presupposing P $\neq$ NP, admitting of null values causes intractability, even in case of $\mathcal{ALC}$. Thus our results suggest that the main source of computational complexity of terminological reasoning seems to be the ability to express *incomplete* knowledge.

5 Description Logics as Tractable Query Languages for Databases

Another interpretation of our results is that, when taken together with the least and greatest fixed point semantics, the universal concept language $\mathcal{U}$ can serve as a powerful but tractable query language for relational databases comprising solely unary and binary relations.[6] From this point of view terminologies are to be thought of as defining so-called *views*, possibly defined recursively.

At this very point, it is important to note that the universal description logic $\mathcal{U}$ is so strong in expressive power that it is even capable of *accurately* defining concepts such as directed acyclic graphs (*DAG*s), trees, or binary trees. The powerful role forming primitives of $\mathcal{U}$ actually admit of plausible and nonrecursive definitions of these concepts. As every finite graph can uniquely be represented by a physical knowledge base in a completely straightforward manner, these concepts provide views which can be used to extract from a huge collection of (connected) directed graphs exactly those which are acyclic or those which are trees or binary trees. If we additionally have recursive concept introductions along with least fixed point semantics at our disposal, we may even extract from a finite and-or-graph G (or a collection of such) exactly the *solvable* vertices, i.e., those vertices which are a root of an acyclic subgraph G_s of G such that every and-vertex of G_s has exactly those edges it has in G and, moreover, every or-vertex has at least one of those edges it has in G. Figure 6 gives the terminology of $\mathcal{U}$ defining all the concepts mentioned in this section, where the recursive concept introduction of *Solvable* should be given least fixed point semantics. This is just to demonstrate that even though the model check-

[4]In this context, a *definition* is meant to be the subterminology of $\mathcal{T}$ which contains exactly those concept introductions which are involved in the recursion.

[5]Technically speaking, in this case deciding $\mathcal{V} \models_{\mathcal{T}} Q$ in $\mathcal{U}$ is co-NP-complete.

[6]Note that unary and binary relations do suffice as far as only object-oriented databases are concerned.

$$
\begin{array}{rcl}
\mathit{DirectedGraph} & \doteq & \forall \mathit{connected}\colon \mathit{Vertex} \\
\mathit{connected} & \doteq & (\mathit{edge} \sqcup \mathit{edge}^{-1})^{*} \\
\mathit{Acyclic} & \doteq & \forall \mathit{connected}\colon(\mathit{edge}^{+} \leq \neg\epsilon) \\
\mathit{DAG} & \doteq & \mathit{DirectedGraph} \sqcap \mathit{Acyclic} \\
\mathit{Tree} & \doteq & \mathit{DAG} \\
 & \sqcap & \forall \mathit{edge}^{*}\colon\exists^{\leq 1}\mathit{edge}^{-1}\colon \mathit{Vertex} \\
\mathit{BinaryTree} & \doteq & \mathit{Tree} \\
 & \sqcap & \forall \mathit{edge}^{*}\colon\exists^{\leq 2}\mathit{edge}\colon \mathit{Vertex} \\
\mathit{AndOrGraph} & \doteq & \mathit{DirectedGraph} \\
 & \sqcap & \forall \mathit{connected}\colon \mathit{AndOrVertex} \\
\mathit{AndOrVertex} & \doteq & \mathit{AndVertex} \sqcap \neg \mathit{OrVertex} \\
 & \sqcup & \mathit{OrVertex} \quad \sqcap \neg \mathit{AndVertex} \\
\mathit{Solvable} & \doteq & \neg\exists \mathit{edge}\colon \mathit{Vertex} \\
 & \sqcup & \mathit{AndVertex} \sqcap \forall \mathit{edge}\colon \mathit{Solvable} \\
 & \sqcup & \mathit{OrVertex} \quad \sqcap \exists \mathit{edge}\colon \mathit{Solvable}
\end{array}
$$

Figure 6: A terminology of $\mathcal{U}$.

ing approach to terminological knowledge representation does make it possible to answer queries in polynomial time, there are actually nontrivial inferences to perform.

Acknowledgements
I would like to thank Martin Buchheit for valuable comments on earlier drafts of the abstract.

References

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

[Chandra and Merlin, 1977] Ashok K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proceedings of the 9th ACM Symposium on Theory of Computing*, pages 77–90, 1977.

[Church, 1936] Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936.

[Emerson and Lei, 1986] E. Allen Emerson and Chin-Laung Lei. Efficient model checking in fragments of the propositional mu-calculus (extended abstract). In *Proceedings of the 1st IEEE Symposium on Logic in Computer Science*, pages 267–278, Boston, Mass., 1986.

[Halpern and Vardi, 1991] Joseph Y. Halpern and Moshe Y. Vardi. Model checking vs. theorem proving: A manifesto. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 325–334, Cambridge, Mass., 1991.

[McCarthy, 1968] John McCarthy. Programs with common sense. In M. Minsky, editor, *Semantic Information Processing*, pages 403–418. MIT Press, Cambridge, Mass., 1968.

[Nebel, 1990] Bernhard Nebel. Terminological Reasoning is Inherently Intractable. *Artificial Intelligence*, 43:235–249, 1990.

[Nebel, 1991] Bernhard Nebel. Terminological cycles: Semantics and computational properties. In J. Sowa, editor, *Formal Aspects of Semantic Networks*, pages 331–361. Morgan Kaufmann, San Mateo, Cal., 1991.

[Patel-Schneider, 1987] Peter F. Patel-Schneider. *Decidable, Logic-Based Knowledge Representation.* PhD thesis, University of Toronto, Toronto, Ont., 1987. Computer Science Department, Technical Report 201/87.

[Reiter, 1984] Raymond Reiter. Towards a logical reconstruction of relational database theory. In M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, editors, *On Conceptual Modeling*, pages 191–233. Springer-Verlag, Berlin, FRG, 1984.

[Schild, 1988] Klaus Schild. Undecidability of subsumption in $\mathcal{U}$. KIT Report 67, Department of Computer Science, Technische Universität Berlin, Berlin, FRG, 1988.

[Schild, 1993] Klaus Schild. Terminological cycles and the propositional μ-calculus. DFKI Research Report RR-93-18, German Research Center for Artificial Intelligence (DFKI), Saarbrücken, FRG, April 1993.

[Schild, 1994a] Klaus Schild. Terminological cycles and the propositional μ-calculus. In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 509–520, Bonn, FRG, 1994.

[Schild, 1994b] Klaus Schild. Tractable reasoning in a universal description logic. DFKI Research Report, German Research Center for Artificial Intelligence (DFKI), Saarbrücken, FRG, 1994. Forthcoming.

[Schmidt-Schauß and Smolka, 1991] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[Schmidt-Schauß, 1989] Manfred Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 421–431, Toronto, Ont., 1989.

[Trahtenbrot, 1963] B. A. Trahtenbrot. Impossibility of an algorithm for the decision problem in finite classes. *American Mathematical Society Translation Series*, 23(2):1–5, 1963.

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