Logic-based Knowledge Representation[*]

Franz Baader
LuFg Theoretical Computer Science, RWTH Aachen
Ahornstraße 55, 52074 Aachen, Germany
e-mail: baader@informatik.rwth-aachen.de

Abstract

After a short analysis of the requirements that a knowledge representation language must satisfy, we introduce Description Logics, Modal Logics, and Nonmonotonic Logics as formalisms for representing, respectively, terminological knowledge, time-dependent and subjective knowledge, and incomplete knowledge. At the end of each section, we briefly comment on the connection to Logic Programming.

Introduction

Let us start with asking under which conditions one may rightfully claim to have *represented knowledge* about an application domain, and not just *stored data* occurring in this domain.[1] One aspect of the requirements on knowledge representation formalisms that can be derived from this consideration is very well satisfied by logical formalisms. We shall see, however, that some other aspects are not treated satisfactorily, at least not by classical first-order predicate logic. Logic Programming languages, as indicated by their name, are *programming* languages, and thus not necessarily appropriate as *representation* languages.

In a nut-shell (and somewhat exaggerated), the difference between knowledge-based programming (which processes knowledge) and classical programming

[*]This is a translation of an article that has appeared (in German) in the journal KI, 3/96:8-16, 1996.

[1]The division between knowledge and data is, of course, not strict, and we do not claim to have a definitive answer to the problem of distinguishing between both.

(which processes data) can be formulated as follows. In classical programming, one designs specialized programs that are tailored to a specific application problem. The knowledge about the problem description and the application domain is implicitly represented in the structure of the program, and must thus be acquired by the programmer. In knowledge-based programming, the knowledge about the application domain is represented explicitly (in an appropriate representation formalism); ideally, the processing can be done with the help of general (i.e., application-independent) problem solving methods. Thus, the knowledge can be acquired and represented by an application expert, who need not be well-acquainted with details of the implementation. As a simple example, let us consider the task of finding out whether two nodes in a directed graph (which may represent the hierarchical organization of a company) are connected (whether one employee is the boss of another one). A first solution, which is very far away from being knowledge-based, might be a program that encodes the structure of the specific graph in its control structure. A second, more reasonable, solution might explicitly represent the graph in an appropriate data structure (which list the nodes that are directly connected), and then code the meaning of "connected" in a program that is independent of the specific graph. This program must be able to compute the connected nodes from the given information about the directly connected nodes. Finally, an even more knowledge-based solution could be one in which the knowledge about the meaning of "connected" is also represented in an explicit way, for example, by the following Horn clauses:

$$\begin{array}{rcl} \text{directly-connected}(x,y) & \rightarrow & \text{connected}(x,y), \\ \text{directly-connected}(x,z) \wedge \text{connected}(z,y) & \rightarrow & \text{connected}(x,y). \end{array}$$

A general problem solving component, which is able to handle such clauses (e.g., the interpreter of a Logic Programming language), could now use these formulae together with the explicit information on the directly connected nodes to infer the connected nodes.

A knowledge representation (KR) formalism should allow for the symbolic representation of all the knowledge relevant in a given application domain. From what we have said so far about the use of knowledge in knowledge-based programming, we can derive two requirements that such a formalism must satisfy. On the one hand, it must be equipped with a *declarative semantics*, that is, the meaning of the entries in a knowledge base (KB) must be defined independent of the programs that operate on the KB (no purely procedural semantics). Usually, such a declarative semantics is given by mapping the symbolic expressions into (an abstraction of) the relevant segment of the "world." In addition, one needs a notion of "truth," which makes it possible to determine which of the symbolic statements hold in the current "world."

On the other hand, one needs an *"intelligent" retrieval mechanism*, which is able to realize the above-mentioned general problem solving component. This mechanism should allow to retrieve knowledge that is only implicitly present in the KB, that is, it should be able to deduce implicitly represented knowledge from the explicit knowledge. The behaviour of this deductive component should depend only on the semantics of the representation language, and not on the syntactic form of the entries in the KB, i.e., semantically equivalent entries should lead to the same results. If we use first-order predicate logic as semantics for the Horn clauses in our above example, then the usual Prolog interpreters do not fulfill this requirement. According to the semantics, the order of the clauses and of the conjunctions is irrelevant. Any Prolog programmer will know, however, that such a re-ordering may drastically change the behaviour of the program.

Another requirement that is usually imposed on KR formalisms is that of allowing for a *structured representation* of the knowledge. This means that semantically related information (for example, all the knowledge about knowledge representation based on Description Logics) should also syntactically be grouped together. This requirement is, on the one hand, justified by cognitive adequacy.[2] On the other hand, there are purely pragmatic reasons, since a structured representation allows for faster retrieval.

Critics of a logic-based approach to KR often (implicitly) equate logic with first-order predicate logic. If we consider in how far *first-order predicate logic* satisfies the requirements introduced above, it shows in fact some strong deficits. The Tarskian semantics of predicate logic is the prototype of a declarative semantics; however, it does not allow for an adequate treatment of incomplete and contradictory knowledge, or of subjective and time-dependent knowledge. Later on, we shall see that this deficit can be overcome by considering Nonmonotonic Logics and Modal Logics. The usual syntax of first-order predicate logic does not support a structured representation of knowledge. Since all the relevant inference problems are undecidable, it is also not possible to provide for semantically adequate inference procedures. In the following section, we shall introduce so-called Description Logics,[3] which are an attempt to overcome both of the last-mentioned problems by using a nonstandard syntax and by restricting the expressive power.

[2]In the human brain, correlated information is also not stored in unrelated parts.

[3]Other names used in the literature are "terminological logics," "concept languages," and "KL-ONE-like KR languages."

According to what we have said until now, Logic Programming languages qualify as representation languages only if they are equipped with an appropriate declarative semantics, which is not the case for Prolog, as illustrated by the example. It should be noted that we do not claim that it is not possible to solve a specific representation problem with the help of a Prolog program: since Prolog is computationally complete, this is always possible. However, this is a programming approach in which the knowledge is not encoded independent of the way in which it is processed. Exactly because of Prolog being computationally complete, all the responsibility (e.g., for termination of the inference process) lies in the hands of programmer, and cannot automatically be guaranteed. In a KR system, the intelligent retrieval mechanisms should (ideally) be able to handle all the knowledge that is represented in a syntactically correct way, which means that one must restrict oneself to a sublanguage (such as Datalog) if one wants to use a Logic Programming approach for KR. An overview of extended Logic Programs with declarative semantics and their application to representing knowledge can be found in [32, 82, 13, 2].

Description Logics

The attempt to provide for a structured representation of information was one of the main motivations for introducing early KR formalisms such as Semantic Networks and Frames. *Frames* have been introduced by Minsky [83] as record-like data structures for representing prototypical situations and objects. The key idea was to collect all the information necessary for treating a situation in one place (the frame for this situation). In [83], Minsky combined his introduction of the frame idea with a general rejection of logic as a KR formalism. Hayes [54, 55] criticized that Frames lack a formal semantics, and showed that with an appropriate formalization (of their monotonic, non-procedural aspects), Frames can be seen as a syntactic variant of first-order predicate logic.

Semantic Networks, which we shall consider in somewhat more detail, have been developed by Quillian [96] for representing the semantics of natural language (this explains their name). They allow to represent concepts and objects as nodes in a graph (see Figure 1). Such a graph has two different types of edges: property edges assign properties (like colour) to concepts (e.g., frog) and objects (e.g., Kermit), whereas IS-A-edges introduce hierarchical relationships among concepts and instance relationships between objects and concepts.

Properties are inherited along IS-A-edges. For example, tree frogs, and thus

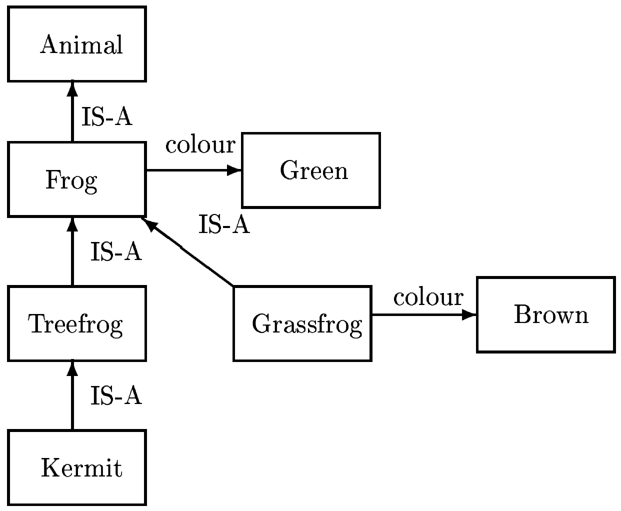

Figure 1: A semantic network.

also Kermit, inherit the colour green from frogs. In many systems, inheritance is only by default, that is, grass frogs do not inherit the property green, since this would be in contradiction with the explicit property edge saying that grass frogs are brown. The missing formal *semantics of Semantic Networks* was criticized by Woods [110] and Brachman [17]. The meaning of a given Semantic Network was left to the intuition of the users and the programmers who implemented the programs processing the networks. Consequently, identical networks could lead to very different results, depending on which of the systems was used to process it. As an illustration of this problem, we point out the possible ambiguities in the interpretation of property edges. In Figure 1, the property edge “colour” from “Frog” to “Green” may, on the one hand, mean that green is the only possible colour for frogs (value restriction). On the other hand, it could mean that any frog has at least the colour green, but may have other colours too (it might be green with red stripes). A partial reconstruction of the semantics of Semantic Networks within first-order predicate logic was presented in [105].

Description Logics (DL) make the quantifier that is implicitly present in property edges (universal in the reading as value restrictions, and existential for the other option) explicit (see below). They descend from so-called “structured inheritance networks” [17, 18], which were first realized in the system KL-ONE [20]. Their main idea is to start with atomic *concepts* (unary predicates) and *roles* (binary predicates), and use a (rather small) set of epistemologically adequate constructors to build complex concepts and roles.

This idea has been further developed both from the theoretical and the practical point of view. In particular, there is a great variety of successors systems (e.g., BACK [90, 92, 59], CLASSIC [16, 19], CRACK [22], K-Rep [77, 78], KRIS [7], LOOM [74, 73], SB-ONE [62]), which have been used in different application domains such as natural language processing [95], configuration of technical systems [111], as software information system [29], for optimizing queries to databases [26], or for support in planning [63].

Figure 2 introduces syntax and semantics of some of the concept constructors employed in systems or investigated in the literature. Most of the systems do not provide for all of these constructors, an vice versa, they may use additional constructors not introduced here. An extensive list of (most of) the constructors considered until now can be found in [4]. The first column

(and $C_1 \ldots C_n$)	$C_1 \sqcap \ldots \sqcap C_n$	$C_1^{\mathcal{I}} \cap \ldots \cap C_n^{\mathcal{I}}$
(or $C_1 \ldots C_n$)	$C_1 \sqcup \ldots \sqcup C_n$	$C_1^{\mathcal{I}} \cup \ldots \cup C_n^{\mathcal{I}}$
(not C)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{I}$
(some R C)	$\exists R.C$	$\{d \in \Delta^{\mathcal{I}} \mid \exists e\colon (d,e) \in R^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}$
(all R C)	$\forall R.C$	$\{d \in \Delta^{\mathcal{I}} \mid \forall e\colon (d,e) \in R^{\mathcal{I}} \Rightarrow e \in C^{\mathcal{I}}\}$
(atleast n R)	$\geq n\, R$	$\{d \in \Delta^{\mathcal{I}} \mid \mathrm{card}(\{e \in \Delta^{\mathcal{I}} \mid (d,e) \in R^{\mathcal{I}}\}) \geq n\}$
(atmost n R)	$\leq n\, R$	$\{d \in \Delta^{\mathcal{I}} \mid \mathrm{card}(\{e \in \Delta^{\mathcal{I}} \mid (d,e) \in R^{\mathcal{I}}\}) \leq n\}$
(and $R_1 \ldots R_n$)	$R_1 \sqcap \ldots \sqcap R_n$	$R_1^{\mathcal{I}} \cap \ldots \cap R_n^{\mathcal{I}}$

Figure 2: Syntax and semantics of concept and role terms.

of the figure shows the (Lisp-like) concrete syntax that is used in most of the systems, whereas the second column introduces the abstract syntax that is usually employed in theoretical work on Description Logics. Starting with atomic concepts and roles, one can use these constructors to build complex *concept terms* and (in the last row) *role terms*. Using the abstract syntax, the concept "Frog," which has been introduced in the Semantic Network of Figure 1, can be described by the concept term $\texttt{Animal} \sqcap \forall\texttt{colour}.\texttt{Green}$, where `Animal` and `Green` are atomic concepts (concept names), and `colour` is an atomic role (role name). The universal quantifier makes clear that we have interpreted the property edge as a value restriction.

In order to define the semantics of concept terms, one considers interpretations $\mathcal{I}$, which consist of a non-empty set $\Delta^{\mathcal{I}}$ (the domain of the interpretation) and an interpretation function that assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The third column in Figure 2 shows how this interpretation function is extended to concept and role terms. An alternative way of defining the semantics of concept and role terms is to give a translation into formulae of first-order predicate logic: concept terms are translated into formulae with one free variable and role terms into formulae with two free variables. The exact definition of this translation is an easy consequence of the third column of Figure 2. For example, the above concept term for the concept "Frog" yields the following formula with free variable x:

$$\mathtt{Animal(x)} \wedge \forall \mathtt{y}{:}(\mathtt{colour(x,y)} \rightarrow \mathtt{Green(y)}).$$

Concept definitions can now be used to assign names (abbreviations) to large terms. For example, the definition $\mathtt{Frog} \doteq \mathtt{Animal} \sqcap \forall\mathtt{colour.Green}$ assigns the name $\mathtt{Frog}$ to the term $\mathtt{Animal} \sqcap \forall\mathtt{colour.Green}$, which can then be used as abbreviation for this term when constructing new concept terms.

A *terminology* (TBox) consists of a finite set of concept and role definitions of the form $A \doteq C$ and $P \doteq R$, where A is a concept name, P is a role name, C is a concept term, and R is a role term. Usually, one imposes the additional requirement that the definitions are unique (i.e., any name may occur at most once as a left-hand side of a definition) and acyclic (i.e., the definition of a name must not, directly or indirectly, refer to this name). As a consequence, definitions can be seen as macros, which can simply be expanded. An interpretation $\mathcal{I}$ is a model of a TBox iff it satisfies all the definitions $A \doteq C$ and $P \doteq R$ contained in the TBox, i.e., if $A^{\mathcal{I}} = C^{\mathcal{I}}$ and $P^{\mathcal{I}} = R^{\mathcal{I}}$ holds for these definitions.

In the *assertional component* of the knowledge base, one can introduce individuals (by giving them names), and assert properties of these individuals. If a, b are names for individuals, C is a concept term, and R a role term, then $C(a)$ and $R(a,b)$ are *assertions*. A finite set of such assertions is called an *ABox*. An interpretation $\mathcal{I}$ (which also assigns elements $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to individual names a) is a model of these assertions iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. For example, $\mathtt{Frog(KERMIT)}$ is a concept assertion and $\mathtt{colour(KERMIT, Colour07)}$ is a role assertion.

A knowledge base (KB) consist of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. As mentioned in the introduction, one is not just interested in retrieving the knowledge that is explicitly stored in such a KB: one should also like to have access to the implicitly represented knowledge. To make this possible, one must be able to draw inferences from the explicitly represented knowledge. As examples of two typical inference problems in Description Logics we shall consider the subsumption and the instance problem. *Subsumption* is concerned with the question whether one concept is a subconcept of another one. Formally, we define for given concept terms C and D and a TBox $\mathcal{T}$: C is subsumed by D w.r.t. $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds in all models $\mathcal{I}$ of $\mathcal{T}$. DL systems usually offer the computation of the subsumption hierarchy of the concepts defined in a TBox as system service (classification). When defining the *instance problem*, one considers a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. The individual a is an instance of the concept term C w.r.t. $\mathcal{T}$ and $\mathcal{A}$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ holds in all models of $\mathcal{T}$ and $\mathcal{A}$. For example, if the TBox contains the above definition of the concept `Frog`, and the ABox contains the above assertions for `KERMIT`, then `COLOUR07` is an instance of `Green` with respect to this TBox and ABox.

Most DL systems (e.g., BACK, CLASSIC, K-REP, LOOM) employ inference procedures that only partially satisfy the requirement that the result should depend only on the semantics of the representation language, and not on the syntactic form of the entries in the KB. These systems use so-called *structural algorithms* (see, e.g., [88], Chapter 4), which are still based on a point of view derived from Semantic Networks: the knowledge base is seen as a directed graph. Structural subsumption algorithms try to detect similarities in the graphs corresponding to the concepts to be tested for subsumption. They have the advantage that they are usually very efficient (polynomial). An important disadvantage is that they are only sound, but *not complete* for reasonably expressive representation languages. If an incomplete subsumption algorithm answers a subsumption query with "yes," then this answer is correct (soundness). If it answers with "no," then this does not mean anything: because of the incompleteness, the subsumption relationship might nevertheless hold. Thus, the behaviour of the algorithm does not depends only on the semantics, but also on other factors that are not transparent to the user.

Since 1988, a new type of algorithms, so-called *tableau-based algorithms*, for reasoning in description logics has been developed. Here, the logical point of view is not only used to define the semantics, but also for the design of algorithms, that is, the inference problems are considered as deduction problems in logics. In principle, these algorithms are methods for generating finite models. They can be seen as specializations of the tableau calculus for first-order predicate logic. The non-standard syntax of DL and the restricted expressiveness of these logics allows to design terminating procedures, i.e., for many description languages one obtains decision procedures for the relevant inference problems (see [8] for an introductory exposition of tableau-based inference methods in DL). The first tableau-based subsumption algorithm was developed in [104] for the language $\mathcal{ALC}$, which allows for the first five constructors of Figure 2. Since then, this approach for designing subsumption algorithms was extended to the instance problem [56, 8] and to various description languages extending $\mathcal{ALC}$ (see, e.g., [58, 57, 11, 12] for languages with number restrictions; [3] for transitive closure of roles and [98] for transitive roles; [6] for constructs that allow to refer to concrete domains such as numbers; and [25] for the treatment of general axioms of the form $C \doteq D$, where C, D may both be complex concept terms).

Other important research contributions for DL are concerned with the decidability and the complexity of the subsumption problem in different DL languages. It has turned out that the languages used in early DL systems were too expressive, which led to undecidability of the subsumption problem [103, 91]. The first complexity results [69, 87] showed that there cannot be polynomial subsumption algorithms, even for very small languages. In the meantime, the worst-case complexity of subsumption in a large class of DL languages has (almost completely) been determined [35, 34, 37]. With the exception of a few polynomially decidable languages, the complexity results range between NP or coNP and PSPACE. Whereas these results are given with respect to an empty TBox (i.e., they consider subsumption of concept terms with respect to all interpretations), Nebel [89] has shown that the expansion of TBox definitions may lead to an exponential blow-up, which may results in a larger complexity (coNP instead of polynomial) for certain languages. The use of general axioms (in $\mathcal{ALC}$) even causes the subsumption problem to become ExpTime-complete [102]. It has also been shown that for certain languages the instance problem can be harder than the subsumption problem [99, 36]. Considering these complexity results, one may ask whether incomplete, but polynomial algorithms should not preferred over the complete ones, which are necessarily of high worst-case complexity. First experiences [5, 22] with implemented systems using complete algorithms show, however, that on realistic KBs the run time is comparable to that of CLASSIC and LOOM (i.e., mature systems using incomplete algorithms).

Before we turn to the connection between DL and Logic Programming, we should like to mention two interesting connections between DL and more traditional areas of logics. Schild [100] was the first to observe that the language $\mathcal{ALC}$ is a syntactic variant of the propositional multi-modal logic $\mathbf{K}_n$ (see next section), and that the extension of $\mathcal{ALC}$ by transitive closure of roles [3] corresponds to propositional dynamic logic (PDL). In particular, the algorithms used in modal logics for deciding satisfiability are very similar to the tableau-based algorithms newly developed for DL languages. This connection between DL and modal logics has been used to transfer decidability results from modal logics to DL [101, 102, 48, 49].

Decidability of the inference problems for $\mathcal{ALC}$ can also be obtained as a consequence of the known decidability result for the two variable fragment of first-order predicate logic. The language L_2 consists of all formulae of first-order predicate logic that can be built with the help of predicate symbols (including equality) and constant symbols (but without function symbols) using only the variables x, y. Decidability of L_2 has been shown in [85]. It is easy to see that, by appropriately re-using variable names, any concept term of the language $\mathcal{ALC}$ can be translated into an L_2-formula with one free variable (see [15] for details). A direct translation of the concept term $\forall R.(\exists R.A)$ yields the formula $\forall y{:}(R(x,y) \rightarrow (\exists z{:}(R(y,z) \wedge A(z))))$. Since the subformula $\exists z{:}(R(y,z) \wedge A(z))$ does not contain x, this variable can be re-used: renaming the bound variable z into x yields the equivalent formula $\forall y{:}(R(x,y) \rightarrow (\exists x{:}(R(y,x) \wedge A(x))))$, which uses only two variables. This connection between $\mathcal{ALC}$ and L_2 shows that any extension of $\mathcal{ALC}$ by constructors that can be expressed with the help of only two variables yields a decidable DL. Number restrictions and composition of roles are examples of constructors that cannot be expressed within L_2. Number restrictions can, however, be expressed in C_2, the extension of L_2 by counting quantifiers, which has recently been shown to be decidable [51].

Since Logic Programming languages are computationally complete and DL languages are usually decidable, one may say that DL languages have less expressive power. If we consider Logic Programming languages as representation languages rather than as programming languages, then one observes that several of the DL constructors cannot be expressed. In fact, disjunction and existential restrictions allow for incompletely specified knowledge. For example, the term

$$(\exists \texttt{pet}.(\texttt{Dog} \sqcup \texttt{Cat}))(\texttt{BILL})$$

leaves it open which ABox individual is Bill's pet, and whether it is a cat or a dog. Such an "under-specification" is not possible with the Horn clauses of traditional Logic Programming languages. To overcome this deficit, extensions of Logic Programming languages by disjunction and classical negation (in contrast to "negation as failure") have been introduced [47, 94, 72, 13, 21]. However, these extensions treat only some aspects of these constructors: for example, the "classical negation" in these approaches only represents the aspect that a set and its complement are disjoint; the fact that the union of a set with its complement is the whole universe is not taken into account. The integration of Description Logics with a rule calculus that is able to express Horn rules has been investigated in [53]. Other work in this direction can, for example, be found in [70].

Modal Logics

This is an area of logics that has been investigated for quite a while, and for which a great variety of methods and results are available. In the following, we give a very short introduction, which emphasizes the connection between Description Logics and Modal Logics. For more detailed introductions and overviews of the area we refer the reader to [27, 60, 52, 39].

The propositional multi-modal logic $\mathbf{K}_n$ extends propositional logic by n pairs of unary operators, which are called box and diamond operators. The $\mathbf{K}$ stand for the basic modal logic on which most modal logics are build (see below), and "multi-modal" means that one considers more than one pair of box and diamond operators. Depending on the intended application, these operators may have different intuitive meanings. For example, if we want to represent time-dependent knowledge, then we can use the diamond operator $\langle \mathit{future} \rangle$ and the box operator $[\mathit{future}]$, where the intended meaning of a formula $\langle \mathit{future} \rangle \phi$ is "Sometime in the future, ϕ holds," whereas $[\mathit{future}]\phi$ is meant to express "Always in the future, ϕ holds." If we want to represent knowledge about the beliefs of intelligent agents, then we can use the operators $\langle \mathit{robi1} \rangle$ and $[\mathit{robi1}]$, where $[\mathit{robi1}]\phi$ should be interpreted as "Robot 1 believes that ϕ holds," and $\langle \mathit{robi1} \rangle \phi$ as "Robot 1 believes that ϕ is possible." The different meanings of the operators are taken into account by using additional axioms or by imposing additional restrictions on the semantic structures (see below).

First, we consider the base logic $\mathbf{K}_n$ in a bit more detail. *Formulae* of this logic are built from atomic propositions p and n different modal parameters m using the Boolean connectives $\wedge, \vee, \neg$[4] and the modal operators $[m]$ und $\langle m \rangle$. For example,

$$[\mathit{robi1}]\langle \mathit{future} \rangle (p \wedge \langle \mathit{robi2} \rangle \neg p)$$

is a formula of $\mathbf{K}_n$, which could be interpreted as saying "Robot 1 believes that sometime in the future, p will hold, while at the same time Robot 2 will believe that $\neg p$ is possible." Here *robi1*, *robi2* and *future* are modal parameters, and p is an atomic proposition.

In order to define the semantics of $\mathbf{K}_n$, we consider so-called *Kripke structures* $\mathcal{K} = (\mathcal{W}, \mathcal{R})$, which consist of a set of possible worlds $\mathcal{W}$ and a set $\mathcal{R}$ of transition relations. Each possible world $I \in \mathcal{W}$ corresponds to an interpretation of propositional logic, i.e., it assigns a truth value $p^I \in \{0, 1\}$ to every atomic proposition p. The set $\mathcal{R}$ contains for every modal parameter m a transition relation $R_m \subseteq \mathcal{W} \times \mathcal{W}$. Validity of a $\mathbf{K}_n$-formula ϕ in the world I of a Kripke structure $\mathcal{K}$ is defined by induction on the structure of $\mathbf{K}_n$-formulae:

[4]Additional Boolean connectives like implication can, as usual, be introduced as abbreviations.

- $\mathcal{K}, I \models p$ iff $p^I = 1$ (for atomic propositions p).
- $\mathcal{K}, I \models \phi \wedge \psi$ iff $\mathcal{K}, I \models \phi$ and $\mathcal{K}, I \models \psi$. The semantics of the other Boolean operators is defined in the usual way.
- $\mathcal{K}, I \models [m]\phi$ iff $\mathcal{K}, J \models \phi$ holds for *all* J such that $(I, J) \in R_m$.
- $\mathcal{K}, I \models \langle m \rangle\phi$ iff $\mathcal{K}, J \models \phi$ holds for *some* J with $(I, J) \in R_m$.

The $\mathbf{K}_n$-formula ϕ is *valid* iff $\mathcal{K}, I \models \phi$ holds for all Kripke structures $\mathcal{K}$ and all worlds I in $\mathcal{K}$.

This definition of the semantics for $\mathbf{K}_n$ looks very similar to the semantics for DL languages. Concept terms C of $\mathcal{ALC}$ can directly be translated into formulae ϕ_C of $\mathbf{K}_n$ by interpreting concept names as atomic propositions and role names as modal parameters. The Boolean connectives of $\mathcal{ALC}$ are simply replaced by the corresponding Boolean connectives of $\mathbf{K}_n$. Universal role restrictions (value restrictions) are replaced by the corresponding box operator, and existential role restrictions are replaced by the corresponding diamond operator. For example, the concept term $\forall R.A \sqcap \exists S.\neg A$ yields the $\mathbf{K}_n$-formula $[R]A \wedge \langle S \rangle \neg A$.

There is an obvious 1–1 correspondence between Kripke structures $\mathcal{K}$ and interpretations $\mathcal{I}$ of $\mathcal{ALC}$: the domain $\Delta^{\mathcal{I}}$ of $\mathcal{I}$ corresponds to the set of possible worlds $\mathcal{W}$, the interpretation $S^{\mathcal{I}}$ of the role name S corresponds to the transition relation R_S, and the interpretation $A^{\mathcal{I}}$ of the concept name A corresponds to the set of worlds in which A has truth value 1. It is easy to show (by induction on the structure of concept terms) that this correspondences also holds for complex concept terms C: if $\mathcal{I}$ is an interpretation of $\mathcal{ALC}$ and $\mathcal{K}$ is the corresponding Kripke structure, then $C^{\mathcal{I}} = \{I \mid \mathcal{K}, I \models \phi_C\}$.

In particular, this implies that $C \sqsubseteq_{\emptyset} D$ iff $\phi_C \rightarrow \phi_D$ is valid. This shows that decision procedures for validity in $\mathbf{K}_n$ can be used to decide the subsumption problem in $\mathcal{ALC}$. One should note, however, that this observation does not yield decision procedures for ABox reasoning, or for reasoning in DL languages with number restrictions. More recent work on modal logics with "graded modalities" [108] (which correspond to number restrictions) and "nominals" [43] (which correspond to ABox individuals) did not focus on decidability issues.

If one wants to assign the modal operators with a specific meaning (like "knowledge of an intelligent agent" or "in the future"), then using the basic modal logic $\mathbf{K}_n$ is not sufficient since it does not model the specific properties that modal operators with this interpretation should satisfy. In order to describe these properties, one can use an axiomatic approach. Figure 3 in-

$$\left.\begin{array}{ll} \text{All propositional tautologies} & \textbf{Taut} \\ [m](\phi \to \psi) \to ([m]\phi \to [m]\psi) & \textbf{K} \\ [m]\phi \to \phi & \textbf{T} \\ [m]\phi \to [m][m]\phi & \textbf{4} \\ \neg[m]\phi \to [m]\neg[m]\phi & \textbf{5} \end{array}\right\} \text{axiom schemata}$$

$$\left.\begin{array}{ll} \phi \to \psi \text{ and } \phi \text{ yield } \psi & \textbf{modus ponens} \\ \phi \text{ yields } [m]\phi & \textbf{necessitation} \end{array}\right\} \text{inference rules}$$

Figure 3: axiom schemata and inference rules for modal logics.

troduces some axiom schemata and the corresponding inference rules modus ponens and necessitation. These are axiom schemata rather than axioms since ϕ and ψ may be substituted by arbitrary modal formulae.

The basic modal logic $\mathbf{K}_n$ is characterized by the axiom schemata **Taut** and **K** in the following sense: a formula ϕ of $\mathbf{K}_n$ is valid (i.e., holds in all worlds of all Kripke structures) iff it can be derived from instances of **Taut** and **K** using modus ponens and necessitation. The other three schemata describe possible properties of modal operators that express "knowledge of intelligent agents," i.e., in this interpretation $[m]\phi$ should be read as "agent m knows ϕ." Thus, **T** can intuitively be read as "An intelligent agent does not have incorrect knowledge," or more precisely: "If agent m knows ϕ in a situation, then ϕ holds in this situation." This property distinguishes knowledge from belief: an agent may very well believe in incorrect facts, but it cannot know them. The axiom schema **4** describes positive introspection, i.e., "An intelligent agent knows what it knows," whereas axiom schema **5** describes negative introspection, i.e., "An intelligent agent knows what it does not know." While **T** and **4** are generally accepted as reasonable axioms for knowledge, **5** is disputable: negative introspection implies that the agent can asses its own competence in the sense that it knows when its own knowledge is not sufficient to solve a certain task. Consequently, there are two different modal logics that model knowledge of intelligent agents. The logic **S4** dispenses with negative introspection, i.e., it uses only the schemata **Taut**, **K**, **T**, **4**, whereas **S5** additionally allows for **5**.

As an alternative to this axiomatic approach for defining **S4** and **S5**, one can also characterize these logics in a semantic way by restricting the admissible transition relations in Kripke structures. The logic **S4** corresponds to the restriction to *reflexive and transitive* transition relations, i.e., the formula

ϕ holds in all worlds of all Kripke structures with reflexive and transitive transition relations iff it can be derived from instances of **Taut**, **K**, **T** and **4** using modus ponens and necessitation. Analogously, **S5** corresponds to the restriction to transition relations that are *equivalence relations*. These correspondences can be used to design tableau algorithms for **S4** and **S5** by integrating the special properties of the transition relations into the rules of the tableau calculus.

The properties of modal operators that model time-dependent knowledge have also been investigated in detail (see, e.g., [38, 107] for overview articles on this topic). In order to represent time-dependent and subjective knowledge in Description Logics, one can integrate modal operators into DL languages. Because of the above mentioned close connection between DL and modal logics, such an integrations appeared to be rather simple. It has turned out, however, that this integration is more complex than expected, both from the semantic and the algorithmic point of view [67, 10, 9, 101].

We close this section by mentioning some work that is concerned with the connection between modal logics and logic programming. Gelfond [44] extends Disjunctive Logic Programs to Epistemic Logic Programs by introducing modal operators **K** and **M**: for a literal L, the expression $\mathbf{K}L$ should be read as "L is known" and $\mathbf{M}L$ should be read as "L may be assumed." Giordano and Martelli [50] use modal logic to obtain a uniform representation of different approaches for structuring Logic Programs with the help of blocks and modules. In [28, 86, 1, 41], modal or temporal Logic Programming languages are introduced.

Nonmonotonic Logics

This research area has also created a huge number of approaches and results, which we cannot describe in detail here. The following is a brief introduction into the existing approaches and the problems treated by these approaches. Overviews of the area can, for example, be found in [42, 23, 24].

Knowledge representation languages based on classical logics (e.g., first-order predicate logic) are monotonic in the following sense: if a statement ϕ can be derived from a knowledge base, then ϕ can also be derived from any larger knowledge base. This property has the advantage that once drawn inferences need not be revised when additional information comes in. However, this leads to the disadvantage that adding information contradicting one of the drawn inferences leads to an inconsistency, and thus makes the knowledge base useless. In many applications, the knowledge about the world (the application domain) is represented in an incomplete way. Nevertheless, one wants to draw *plausible* conclusions from the available knowledge. In this situation, newly acquired information may show that some of these plausible conclusions were wrong. This should not lead to inconsistency of the knowledge base, but rather to a withdrawal of some of the plausible conclusions.

Let us start with three simple examples that illustrate the situations in which nonmonotonic inference methods are desirable. *Default rules* apply to most individuals (resp. typical or normal individuals), but not to all. As an example, we may consider property edges in Semantic Networks. In the network of Figure 1 we had the default rule "Frogs are *normally* green." This rule should be applied whenever there is no information contradicting it. As long as we only know that Kermit is a frog, we deduce that Kermit is green. The rule is not applied to grass frogs, of which it is known that they are not green (since they are brown).

The *Closed World Assumption (CWA)* assumes by default that the available information is complete. If an assertion cannot be derived (using classical inference methods) from the knowledge base, then CWA deduces its negation. This assumption is, for example, employed in relational databases and in Logic Programming languages with "Negation as Failure." As an example, we may consider train connections in a timetable: if a connection is not contained in the timetable, we conclude that it does not exist. If we learn later on that there is such a connection (which was not contained in the timetable), then we must withdraw this conclusion.

The *Frame problem* comes from the domain of modelling actions. In this context, it is important to describe which properties are changed and which remain unchanged by an application of the action. Usually, the application of an action changes very few aspects of the world. For example, by sending a manuscript of this article to a publisher, I have changed its location, but (hopefully) not its content. The so-called "frame axioms" describe, which properties remain unchanged by the application of an action. Since there usually is a very large number of these axioms, one should try to avoid having to state the explicitly. A possible solution to this problem is to employ a nonmonotonic inference rule, which says that (normally) all aspects of the world that are not explicitly changed by the action remain invariant under its application.

In the literature, a great variety of different approaches to nonmonotonic reasoning have been introduced, of which none seems to be "the best" approach. A very positive development is the fact that recently several results clarifying the connection between different approaches have been obtained (see, e.g., [61, 64, 65, 71, 66]). In the following, we briefly introduce the four most important types of approaches.

Consistency-based approaches, of which Reiter's Default Logic [97] is a typical example, consider nonmonotonic rules of the form "A normally implies B." Such a rule can be applied (i.e., B is inserted in the knowledge base), if A holds, and inserting B does not destroy consistency of the knowledge base. As an example, we may again use the default rule "Frogs are normally green." Let us first assume that the knowledge base contains the information: "Grass frogs are brown. An individual cannot be both brown and green. Grass frogs are frogs. Kermit is a frog. Scooter is a grass frog." In this case, the default rule can be applied to Kermit, but not to Scooter. The major problem that a consistency-based approach must solve is the question of how to resolve conflicts between different rules: applying default rules in different order may lead to different results. To illustrate this problem, let us assume that the strict information "Grass frogs are frogs" is replaced by the defeasible information "Grass frogs are normally frogs," whereas all the other information remains the same. Now, both default rules are applicable to Scooter. As soon as one of them is applied, the other one is no longer applicable. Thus, one must decide which of the possible results should be preferred, or, if the conflict cannot be resolved due to priorities among rules, how much information can still be deduced from such an unresolved situation. For example, the conclusions concerning Kermit should not be influenced by the conflict for Scooter.

A *modal nonmonotonic logic* was, for example, proposed by McDermott and Doyle [81, 80]. This logic allows for a diamond operator, which is written as $\mathbf{M}$, where $\mathbf{M}\phi$ should intuitively be read as "ϕ is consistent." The default rule "Frogs are normally green" can then be written as the implication $\mathtt{Frog} \wedge \mathbf{M}\mathtt{Green} \rightarrow \mathtt{Green}$. In order to treat this implication correctly, the logic needs an additional inference rule, which is able to derive formulae of the form $\mathbf{M}\phi$ according to their intended semantics. In principle, this inference rule allows us to deduce $\mathbf{M}\phi$, if $\neg\phi$ cannot be deduced. This is not as simple as it may sound since we are faced with the following cyclic dependency: which formulae of the form $\neg\phi$ are deducible already depends on the inference rule to be defined. Doyle and McDermott solve this problem by introducing an appropriated fixed-point semantics. Another representative of the class of modal nonmonotonic logics is the well-known autoepistemic logic [84].

In classical predicate logic, the notion of logical consequence is defined with respect to *all models* of a set of formulae. *Preferential semantics* [106] takes as logical consequences all the formulae that hold in *all preferred models*. The preferred models are usually defined as the minimal models with respect to a given *preference relation* on interpretations. To illustrate this idea, we consider a simple example from propositional logic. If we assume that there are only two propositional variables p, q, then we have four different interpretations: $I_{\neg p,\neg q}$, in which p and q are false, $I_{p,\neg q}$, in which p is true and q is false, $I_{\neg p,q}$, in which p is false and q is true, and $I_{p,q}$, in which p and q are true. Assume that the preference relation $<$ is given by $I_{\neg p,\neg q} < I_{p,q}$, $I_{\neg p,\neg q} < I_{\neg p,q}$ $I_{p,q} < I_{p,\neg q}$ and $I_{\neg p,q} < I_{p,\neg q}$. The empty set of formulae $\emptyset$ has all interpretations as model, which means that $I_{\neg p,\neg q}$ is the only minimal model. In particular, $\neg p$ is a consequence of $\emptyset$ with respect to this preferential semantics. The set $\{q\}$ exclude the two interpretations $I_{p,\neg q}$ and $I_{\neg p,\neg q}$. The remaining models, $I_{p,q}, I_{\neg p,q}$ are incomparable w.r.t. $<$, and are thus both minimal models. Since $\neg p$ does not hold in both, it can no longer be deduced. This shows that preferential semantics yields a nonmonotonic formalism. An important example for preferential semantics for the case of predicate logic is *circumscription* [79]. Here, the goal is to minimize the extension of a given predicate P, i.e., an interpretation $\mathcal{I}$ is preferred over an interpretation $\mathcal{J}$, if $P^{\mathcal{I}} \subset P^{\mathcal{J}}$ holds. Default rules like "Frogs are normally green" can then be expressed with the help of an "abnormality predicate," which is minimized:

$$\texttt{Frog(x)} \wedge \neg\texttt{Ab(x)} \rightarrow \texttt{Green(x)}.$$

Exceptions to this rule can now simply be expressed by implications like `Brown(x)` $\rightarrow$ `Ab(x)`. The fact that `Ab` is minimized makes sure that the default rule is applied to an individual unless it is a known exception.

The introduction of *properties* that a "reasonable" *nonmonotonic inference relation* $\mid\!\sim$ should satisfy was, on the one hand, motivated by the fact that such properties can be used to compare and evaluate different approaches to nonmonotonic reasoning [40, 75]. On the other hand, these properties can also be interpreted as inference rules (like modus ponens), which can be used to generate new nonmonotonic consequences [68]. It has also turned out that there is a close connection between preferential semantics and inferences relations satisfying certain properties [66]. Figure 4 gives several examples of such reasonable properties.

A similar approach for evaluating and comparing the semantics for Logic Programs was used in [30, 31]. As mentioned above, the "Closed World Assumption" in Logic Programs, and the corresponding treatment of negation as "Negation as Failure" leads to a nonmonotonic behaviour of Logic Programs. Thus, it is not surprising that there is a close connection between approaches for defining declarative semantics for (extended) logic programs (e.g., [45, 46, 14, 109]) and formalisms for nonmonotonic reasoning. In principle, these semantics depend on a preference relation between models. Their development was strongly influenced by the semantics for autoepistemic logic

$\phi \mathrel{|\!\sim} \phi$		**Reflexivity**
If $\phi \mathrel{|\!\sim} \psi$ and ϕ equivalent to ϕ'	then $\phi' \mathrel{|\!\sim} \psi$	**Left equivalence**
If $\phi \mathrel{|\!\sim} \psi$ and ψ implies ψ'	then $\phi \mathrel{|\!\sim} \psi'$	**Right weakening**
If $\phi \wedge \phi' \mathrel{|\!\sim} \psi$ and $\phi \mathrel{|\!\sim} \phi'$	then $\phi \mathrel{|\!\sim} \psi$	**Cut**
If $\phi \mathrel{|\!\sim} \phi'$ and $\phi \mathrel{|\!\sim} \psi$	then $\phi \wedge \phi' \mathrel{|\!\sim} \psi$	**Cautious monotony**

Figure 4: Properties of nonmonotonic inference relations.

and for default logic. A first overview of these connections is given in [93]. More recent work in this direction can be found in the proceedings of the conferences "Non-Monotonic Extensions of Logic Programming" [33] and "Logic Programming and Nonmonotonic Reasoning" [76].

References

[1] M. Abadi and Z. Manna. Temporal logic programming. *Journal of Symbolic Computation*, 8:277–295, 1989.

[2] K. R. Apt and R. N. Bol. Logic programming and negation: A survey. *Journal of Logic Programming*, 19 & 20:9–71, 1994.

[3] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 446–451, Sydney, Australia, 1991.

[4] F. Baader, H.-J. Bürckert, J. Heinsohn, J. Müller, B. Hollunder, B. Nebel, W. Nutt, and H.-J. Profitlich. Terminological knowledge representation: A proposal for a terminological logic. DFKI Technical Memo TM-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz, Kaiserslautern, 1990.

[5] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological systems. *Journal of Applied Intelligence*, 4:109–132, 1994.

[6] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 452–457, Sydney, Australia, 1991.

[7] F. Baader and B. Hollunder. $\mathcal{KRIS}$: $\mathcal{K}$nowledge $\mathcal{R}$epresentation and $\mathcal{I}$nference $\mathcal{S}$ystem. *SIGART Bulletin*, 2(3):8–14, 1991.

[8] F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithms. In M. Richter and H. Boley, editors, *Proceedings of the First International Workshop on Processing Declarative Knowledge*, volume 567 of *Lecture Notes in Computer Science*, pages 67–85, Kaiserslautern (Germany), 1991. Springer–Verlag.

[9] F. Baader and A. Laux. Terminological logics with modal operators. In C. Mellish, editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 808–814, Montréal, Canada, 1995. Morgan Kaufmann.

[10] F. Baader and H.-J. Ohlbach. A multi-dimensional terminological knowledge representation language. *Journal of Applied Non-Classical Logics*, 5(2):153–197, 1995.

[11] F. Baader and U. Sattler. Description logics with symbolic number restrictions. In W. Wahlster, editor, *Proceedings of the Twelfth European Conference on Artificial Intelligence (ECAI-96)*, pages 283–287. John Wiley & Sons Ltd, 1996.

[12] F. Baader and U. Sattler. Number restrictions on complex roles in description logics. In *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning (KR-96)*. Morgan Kaufmann, Los Altos, 1996.

[13] C. Baral and M. Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 19 & 20:73–148, 1994.

[14] N. Bidoit and C. Froidevaux. Negation by default and unstratified logic programs. *Theoretical Computer Science*, 78:85–112, 1991.

[15] A. Borgida. On the relationship between description logic and predicate logic queries. Research Report LCSR-TR-295-A, Department of Computer Science, Rutgers University, Piscataway, NJ, USA, 1993.

[16] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: A structural data model for objects. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 59–67, Portland, Oreg., 1989.

[17] R. J. Brachman. What's in a concept: Structural foundations for semantic networks. *International Journal of Man-Machine Studies*, 9:127–152, 1977.

[18] R. J. Brachman. Structured inheritance networks. In W. A. Woods and R. J. Brachman, editors, *Research in Natural Language Understanding*, Quarterly Progress Report No. 1, BBN Report No. 3742, pages 36–78. Bolt, Beranek and Newman Inc., Cambridge, Mass., 1978.

[19] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In J. Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, San Mateo, Calif., 1991.

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

[21] S. Brass and J. Dix. Disjunctive semantics based upon partial and bottom-up evaluation. In L. Sterling, editor, *Proceedings of the 12th International Conference on Logic Programming*, pages 199–213. MIT Press, 1995.

[22] P. Bresciani, E. Franconi, and S. Tessaris. Implementing and testing expressive description logics: A preliminary report. In *Proceedings of the International Symposium on Knowledge Retrieval, Use, and Storage for Efficiency, KRUSE-95, Santa Cruz, USA*, Lecture Notes in Artificial Intelligence. Springer–Verlag, 1996. To appear.

[23] G. Brewka. *Nonmonotonic Reasoning: Logical Foundations of Commonsense.* Cambridge University Press, Cambridge, UK, 1991.

[24] G. Brewka, J. Dix, and K. Konolige. *Nonmonotonic Reasoning: An Overview.* CSLI Publications, Center for the Study of Language and Information, Stanford, Cal., 1997.

[25] M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence, IJCAI'93*, pages 704–709, Chambery, France, 1993.

[26] M. Buchheit, M. Jeusfeld, W. Nutt, and M. Staudt. Subsumption of queries to object-oriented databases. *Information Systems*, 19(1):33–54, 1994.

[27] B. F. Chellas. *Modal Logic: An Introduction.* Cambridge University Press, Cambridge, UK, 1980.

[28] L. Farinãs del Cerro. Molog: A system that extends Prolog with modal logic. *New Generation Computing*, 4:35–50, 1986.

[29] P. Devanbu, R. J. Brachman, P. G. Selfridge, and B. W. Ballard. LaSSIE: A knowledge-based software information system. *Communications of the ACM*, 34(5):34–49, 1991.

[30] J. Dix. A classification-theory of semantics of normal logic programs: I. Strong properties. *Fundamenta Informaticae*, XXII(3):227–255, 1995.

[31] J. Dix. A classification-theory of semantics of normal logic programs: II. Weak properties. *Fundamenta Informaticae*, XXII(3):257–288, 1995.

[32] J. Dix. Semantics of logic programs: Their intuitions and formal properties. An overview. In A. Fuhrmann and H. Rott, editors, *Logic, Action and Information – Essays on Logic in Philosophy and Artificial Intelligence*, pages 241–327. DeGruyter, 1995.

[33] J. Dix, L. Pereira, and T. Przymusinski, editors. *Non-Monotonic Extensions of Logic Programming, Proceedings*, volume 927 of *Lecture Notes in AI.* Springer–Verlag, 1995.

[34] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 151–162, Cambridge, Mass., 1991.

[35] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 458–463, Sydney, Australia, 1991.

[36] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: From subsumption to instance checking. *Journal of Logic and Computation*, 4(4):423–452, 1994.

[37] F.M. Donini, B. Hollunder, M. Lenzerini, A.M. Spaccamela, D. Nardi, and W. Nutt. The complexity of existential quantification in concept languages. *Journal of Artificial Intelligence*, 53:309–327, 1992.

[38] W. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B.* Elsevier, Amsterdam, The Netherlands, 1990.

[39] M. Fitting. Basic modal logic. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 1.* Oxford University Press, Oxford, UK, 1993.

[40] D. M. Gabbay. Theoretical foundations for non-monotonic reasoning in expert systems. In K. R. Apt, editor, *Proceedings of the NATO Advance Study Institute on Logics and Models of Concurrent Systems, La Colle-sur-Loup, France*, Berlin, 1985. Springer–Verlag.

[41] D. M. Gabbay. Modal and temporal logic programming. In A. Galton, editor, *Temporal Logics and Their Applications.* Academic Press, London, UK, 1987.

[42] D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors. *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3: Nonmonotonic Reasoning and Uncertain Reasoning.* Oxford University Press, Oxford, UK, 1994.

[43] G. Gargov and V. Goranko. Modal logic with names. *J. Philosophical Logic*, 22:607–636, 1993.

[44] M. Gelfond. Logic programming and reasoning with incomplete information. *Annals of Mathematics and Artificial Intelligence*, 12, 1994.

[45] M. Gelfond and V. Lifschitz. The stable semantics for logic programs. In R. Kowalski and K. Bowen, editors, *Proceedings of the 5th International Symposium on Logic Programming*, pages 1070–1080, Cambridge, MA., 1988. MIT Press.

[46] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In D. Warren and P. Szeredi, editors, *Proceedings of the 7th International Conference on Logic Programming*, pages 579–597, Cambridge, MA, 1990. MIT Press.

[47] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[48] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the Twelfth National Conference on Artificial Intelligence, AAAI-94*, pages 205–212. AAAI-Press/The MIT-Press, 1994.

[49] G. De Giacomo and M. Lenzerini. Concept languages with number restrictions and fixpoints, and its relationship with mu-calculus. In *Proceedings of the Eleventh European Conference on Artificial Intelligence, ECAI-94*, pages 411–415. John Wiley and Sons, 1994.

[50] L. Giordano and A. Martelli. Structuring logic programs: A modal approach. *Journal of Logic Programming*, 21(2):59–94, 1994.

[51] E. Grädel, M. Otto, and E. Rosen. Two-Variable Logic with Counting is Decidable. To appear in Proc. LICS'97, 1997.

[52] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logic of knowledge and belief. *Journal of Artificial Intelligence*, 54:319–379, 1992.

[53] P. Hanschke. *A Declarative Integration of Terminological, Constraint-based, Data-driven, and Goal-directed Reasoning.* Infix, Sankt Augustin, 1996.

[54] P. J. Hayes. In defence of logic. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence, IJCAI'77*, pages 559–565, Cambridge, Mass., 1977.

[55] P. J. Hayes. The logic of frames. In D. Mentzing, editor, *Frame Conceptions and Text Understanding.* Walter de Gruyter and Co., Berlin, Germany, 1979.

[56] B. Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In *14th German Workshop on Artificial Intelligence*, volume 251 of *Informatik-Fachberichte*, pages 38–47, Ebingerfeld, Germany, 1990. Springer–Verlag.

[57] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 335–346, Cambridge, Mass., 1991.

[58] B. Hollunder, W. Nutt, and M. Schmidt-Schauß. Subsumption algorithms for concept description languages. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 348–353, Stockholm, Sweden, 1990.

[59] T. Hoppe, C. Kindermann, J. Quantz, A. Schmiedel, and M. Fischer. BACK V5: tutorial and manual. KIT Report 100, Technical University of Berlin, Berlin, Germany, 1993.

[60] G. E. Hughes and M. J. Cresswell. *A Companion to Modal Logic.* Methuen & Co., London, 1984.

[61] T. Imielinski. Results on translating defaults to circumscription. *Journal of Artificial Intelligence*, 32(1):131–146, 1987.

[62] A. Kobsa. First experiences with the SB-ONE knowledge representation workbench in natural language applications. *SIGART Bulletin*, 2(3):70–76, 1991.

[63] J. Koehler. An application of terminological logics to case-based reasoning. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning, KR'94*, pages 351–362, Bonn, Germany, 1994. Morgan Kaufmann.

[64] K. Konolige. On the relation between default and autoepistemic logic. *Journal of Artificial Intelligence*, 35(3):343–382, 1988.

[65] K. Konolige. On the relation between circumscription and autoepistemic logic. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence, IJCAI-89*, pages 1213–1218, San Mateo, CA, 1989. Morgan Kaufmann.

[66] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Journal of Artificial Intelligence*, 44(1–2):167–207, 1990.

[67] A. Laux. Beliefs in multi-agent worlds: A terminological logics approach. In *Proceedings of the Eleventh European Conference on Artificial Intelligence, ECAI-94*. John Wiley and Sons, 1994.

[68] D. Lehmann. What a conditional knowledge base entails. In R. J. Brachman, editor, *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 212–222, Toronto, Ont., 1989.

[69] H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.

[70] A. Levy and M.C. Rousset. CARIN: A representation language combining horn rules and description logics. In W. Wahlster, editor, *Proceedings of the Twelfth European Conference on Artificial Intelligence (ECAI-96)*, pages 323–327. John Wiley & Sons Ltd, 1996.

[71] V. Lifschitz. Between circumscription and autoepistemic logic. In R. J. Brachman, editor, *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 235–244, Toronto, Ont., 1989.

[72] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming.* MIT-Press, 1992.

[73] R. MacGregor. Inside the LOOM classifier. *SIGART Bulletin*, 2(3):88–92, 1991.

[74] R. MacGregor and R. Bates. The LOOM knowledge representation language. Technical Report ISI/RS-87-188, University of Southern California, 1987.

[75] D. Makinson. General patterns in nonmonotonic reasoning. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3: Nonmonotonic and Uncertain Reasoning*, pages 35–110. Oxford University Press, Oxford, UK, 1994.

[76] V. Marek, A. Nerode, and M. Truszczynski, editors. *Logic Programming and Nonmonotonic Reasoning, Third International Conference*, volume 928 of *Lecture Notes in AI.* Springer Verlag, 1995.

[77] E. Mays, C. Apté, J. Griesmer, and J. Kastner. Experience with K-Rep: An object-centered knowledge representation language. In *Proceedings of IEEE CAIA-88*, pages 62–67, 1988.

[78] E. Mays, R. Dionne, and R. Weida. K-Rep system overview. *SIGART Bulletin*, 2(3):93–97, 1991.

[79] J. McCarthy. Circumscription – A form of non-monotonic reasoning. *Journal of Artificial Intelligence*, 13(1 & 2):27–39, 1980.

[80] D. McDermott. Nonmonotonic logic II: Nonmonotonic modal theories. *Journal of the ACM*, 29(1):33–57, 1982.

[81] D. McDermott and J. Doyle. Non-monotonic logic I. *Journal of Artificial Intelligence*, 13(1–2):41–72, 1980.

[82] J. Minker. An overview of nonmonotonic reasoning and logic programming. *Journal of Logic Programming*, 17:95–126, 1993.

[83] M. Minsky. A framework for representing knowledge. In P. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, New York, 1975.

[84] R. C. Moore. Semantical considerations on nonmonotonic logic. *Journal of Artificial Intelligence*, 25(1):75–94, 1985.

[85] M. Mortimer. On languages with two variables. *Zeitschr. f. math. Logik und Grundlagen d. Math.*, 21:135–140, 1975.

[86] B. C. Moszkowski. *Executing Temporal Logic Programs.* Cambridge University Press, Cambridge, UK, 1986.

[87] B. Nebel. Computational complexity of terminological reasoning in BACK. *Journal of Artificial Intelligence*, 34(3):371–383, 1988.

[88] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Computer Science.* Springer–Verlag, 1990.

[89] B. Nebel. Terminological reasoning is inherently intractable. *Journal of Artificial Intelligence*, 43(2):235–249, 1990.

[90] B. Nebel and K. von Luck. Hybrid reasoning in BACK. In Z. W. Ras and L. Saitta, editors, *Methodologies for Intelligent Systems*, volume 3, pages 260–269. North-Holland, 1988.

[91] P. F. Patel-Schneider. Undecidability of subsumption in NIKL. *Journal of Artificial Intelligence*, 39(2):263–272, 1989.

[92] C. Peltason. The BACK system – an overview. *SIGART Bulletin*, 2(3):114–119, 1991.

[93] T. Przymusinski. Non-monotonic formalisms and logic programming. In *Proceedings of the 6th International Conference on Logic Programming*, pages 655–674, Cambridge, MA, 1989. MIT Press.

[94] T. Przymusinski. Stable semantics for disjunctive programs. *New Generation Computing*, 9:401–424, 1991.

[95] J. Quantz and B. Schmitz. Knowledge-based disambiguation for machine translation. *Minds and Machines*, 4:39–57, 1994.

[96] M. Quillian. Semantic memory. In M. Minsky, editor, *Semantic Information Processing*, pages 216–270. MIT Press, Cambridge, Mass., 1968.

[97] Raymond Reiter. A logic for default reasoning. *Journal of Artificial Intelligence*, 13:81–132, 1980.

[98] U. Sattler. A concept language extended with different kinds of transitive roles. In G. Görz and S. Hölldobler, editors, *20. Deutsche Jahrestagung für Künstliche Intelligenz, KI'97*, number 1137 in Lecture Notes in Artificial Intelligence. Springer Verlag, 1996.

[99] A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems*, 2:265–278, 1993.

[100] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 466–471, Sydney, Australia, 1991.

[101] K. Schild. Combining terminological logics with tense logic. In *Proceedings of the 6th Portuguese Conference on Artificial Intelligence, EPIA-93*, pages 105–120, 1993.

[102] K. Schild. Terminological cycles and the propositional μ-calculus. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning, KR'94*, pages 509–520, Bonn, Germany, 1994. Morgan Kaufmann.

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

[104] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Journal of Artificial Intelligence*, 47:1–26, 1991.

[105] L. K. Schubert, R. G. Goebel, and N. J. Cercone. The structure and organization of a semantic net for comprehension and inference. In N. V. Findler, editor, *Associative Networks: Representation and Use of Knowledge by Computers*, pages 121–175. Academic Press, 1979.

[106] Y. Shoham. A semantical approach to nonmonotonic logics. In *IEEE Symposium on Logic in Computer Science*, pages 275–279, Ithaca, N.Y., 1987.

[107] C. Stirling. Modal and temporal logic. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science, Vol. 2.* Oxford University Press, Oxford, UK, 1993.

[108] W. van der Hoek and M. de Rijke. Counting objects. *J. Logic and Computation*, 5(3):325–345, 1995.

[109] A. van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38:620–650, 1991.

[110] W. A. Woods. What's in a link: foundations for semantic networks. In D. G. Bobrow and A. M. Collins, editors, *Representation and Understanding: Studies in Cognitive Science*, pages 35–82. Academic Press, London, 1975.

[111] J. R. Wright, E. S. Weixelbaum, K. Brown, G. T. Vesonder, S. R. Palmer, J. I. Berman, and H. H. Moore. A knowledge-based configurator that supports sales, engineering, and manufacturing at AT&T network systems. *AI Magazine*, 14(3):69–80, 1993.