This is a draft version of a paper to appear in G. Brewka, ed., *Essentials in Knowledge Representation.*

# Artificial Intelligence: A Computational Perspective*

Bernhard Nebel
DFKI, Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
*and*
Fakultät für Informatik, Universität Ulm, D-89069 Ulm, Germany
e-mail: nebel@informatik.uni-ulm.de

November 1994

**Abstract**

Although the computational perspective on cognitive tasks has always played a major role in Artificial Intelligence, the interest in the precise determination of the computational costs that are required for solving typical AI problems has grown only recently. In this paper, we will describe what insights a computational complexity analysis can provide and what methods are available to deal with the complexity problem.

*This work was partially supported by the European Commission as part of DRUMS-II, the ESPRIT Basic Research Project P6156.

# 1 Introduction

It is well-known that typical AI problems, such as natural language understanding, scene interpretation, planning, configuration, or diagnosis are computationally difficult. Hence, it seems to be worthless to analyze the computational complexity of these problems. In fact, some people believe that all AI problems are NP-hard or even undecidable.

Conceiving AI as a scientific field that has as its goal the analysis and synthesis of cognitive processes using the metaphor of computation, it appears to be obvious to exploit the theory of computer science for analyzing AI problems. Interpreting cognitive processes as computational processes implies that the limitations of computational processes apply to cognitive processes as well. In particular, computational complexity theory may be used in order to derive conjectures about the solvability of a given problem. This also holds for the more technical point of view when we are interested in emulating cognitive capabilities using computational processes.

In addition to a feasibility analysis, computational complexity theory can also be used to give a deeper understanding of the structure of a problem. It provides us with insights about the possible sources of complexity and can be used to direct the search for efficient methods to solve the problem, perhaps giving up on generality or the quality of the solution.

In an AI textbook [Rich and Knight, 1991, p. 44], AI is defined as

> the study of techniques for solving exponentially hard problems in polynomial time by exploiting knowledge about the problem domain.

This statement is, of course, a contradiction in terms since an exponentially hard problem cannot be solved in polynomial time by definition. This contradiction can be resolved by noting that the AI notion of what a solution is differs significantly from the standard notion. In AI, one is satisfied with solutions that *almost always* give an *almost correct* answer. In other words, in AI one usually does not solve the original problem but a weaker one [Bylander, 1991]. Before giving up on an accurate solution, however, one should try to find out whether the problem is indeed computationally difficult. Further, if one adapts a weaker problem definition, one should spell that out and try to determine the implied computational difficulty. However, how do we determine the difficulty of a problem?

Complexity theory provides us with the right tools to judge the difficulty of a problem. If we are able to show that a problem can be solved in polynomial time, then it does not make much sense to think about an approximation of the solution. Conversely, if we can show that a problem is NP-hard, then we know that it is very unlikely that we can solve the general problem in reasonable time. However, having shown NP-hardness does usually not mean that we have to give up on the problem completely. To the contrary, such a proof can be viewed as a license for abandoning the search for efficient, complete and accurate methods, and for looking instead for ways to solve interesting aspects by whatever methods are available.

The rest of the paper is structured as follows. In the next section, the reader is reminded of the basic assumptions of complexity theory. Based on that, different kinds of complexity analyses are described in Section 3. In particular, *polynomial special cases*, *classification of all special cases*, *differential analyses* for comparing different formalizations of a given informal problem, and the use of complexity theory to probe for *plausibility of hypotheses* concerning computational issues will be considered. Furthermore, the usefulness of the precise determination of the complexity for NP-hard problems is discussed. In Section 4, some ways to cope with the complexity problem are sketched, such as natural and enforced *restrictions of the problem*, *approximations*, *redifinitions* of the problem, and *heuristics*. Finally, in Section 5, the main points are summarized.

# 2 Basic Assumptions of Computational Complexity Theory

Computational complexity theory[1] tries to classify problems according to their requirements on computational resources (time, memory) depending on the size of the input. In this context, the term *problem* means, contrary to the ordinary meaning, "generic question," which has different *instances*. For example, a particular crossword puzzle is not a problem, but an instance of the problem of crossword puzzle solving.

[1]Good text books on this topic are [Balcázar *et al.*, 1988; Balcázar *et al.*, 1990; Garey and Johnson, 1979; Papadimitriou, 1994].

## 2.1 Polynomial vs. Exponential Runtime Requirements

A problem is considered to be *efficiently solvable* if for all instances of the problem an algorithm can find the correct answer in a number of computation steps that is *polynomially bounded* by the size of the instance – assuming a "traditional" model of computation, i.e., a *sequential, deterministic* computation model, e.g., Turing machines or random access machines.[2]

In computational complexity theory, one usually restricts the attention to so-called *decision problems*, i.e., problems that have only "yes" and "no" as possible answers. Often these problems are also viewed as formal languages formed by the "yes" instances. While this might seem to be a serious restriction at first sight, it turns out that in most cases a polynomial algorithm for a decision problem can be easily transformed into an efficient algorithm for the corresponding general problem.

The class of decision problems that can be characterized by being solvable in polynomial time is denoted by $\mathsf{P}$. Decision problems not belonging to this class, e.g., problems that need exponential time, are considered to be not efficiently solvable. The reason for this judgement is that the growth rate of exponential functions leads to astronomical runtime requirements even for moderately sized instances.

The distinction between polynomial and exponential runtime requirements becomes even more vivid if one considers the effects of further advances in computer technology. Assuming we have a problem $M_1$ that needs $n^2$ steps and that can be solved in reasonable time, e.g. one minute, up to an instance size of $m_1$, then an increase of speed by $10^6$ leads to the effect that instances up to a size of $1000 \times m_1$ can be solved in one minute. For problem $M_2$ that needs $2^n$ steps the picture is quite different, however. Assuming that instances up to a size $m_2$ can be solved with the traditional technology in reasonable time, an increase by $10^6$ in speed leads only to $20 + m_2$ as the maximal instance size.

Although the distinction between "efficiently" and "not efficiently" solvable problems according to whether the number of computation steps can be polynomially bounded or not seems to be reasonable, one should always

[2]Note that the definition of polynomially bounded computation is independent from the particular machine model since all sequential, deterministic machines can be simulated on each other in polynomial time.

keep in mind that it is only a mathematical abstraction. It is based on the assumption that in case of polynomial runtime the exponent is small, and that in the case of exponential runtime the worst case occurs significantly often. As experience shows, however, these assumptions are valid for most naturally occurring problems.

In order to find out, whether a problem is efficiently solvable or not, we only have to find a polynomial algorithm or to prove that such an algorithm is impossible. However, there exist a large number of problems for which no polynomial algorithms are known but at the same time it seems to be impossible to prove that super-polynomial time is necessary to solve these problems. The formal classification of these problems is one of the challenges of complexity theory.

## 2.2 Nondeterministic Computations and the Complexity Class NP

One formal tool to characterize these problems is the *nondeterministic* Turing machine. Such a machine can choose nondeterministically among different successor states during its computation and it accepts an input (answers "yes") if there exists a sequence of nondeterministic choices that leads to an accepting state. A nondeterministic Turing machine accepts a language $L$ in polynomial time if, and only if, all words of $L$ are accepted using only a polynomial number of computation steps on the nondeterministic machine. The class of languages (or decision problems) that are accepted on nondeterministic Turing machines using polynomial time is called NP. Since all deterministic machines can be viewed as nondeterministic machines, it follows that $\mathsf{P} \subseteq \mathsf{NP}$. Whether the converse inclusion holds is an open problem, however. This is the famous $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem.

Although we don't know whether $\mathsf{P} \neq \mathsf{NP}$, it is nevertheless possible to identify the "hardest" problems in NP. The formal tool for doing so are resource-limited reductions between problems. A problem $A$ can be *polynomially many-one reduced* to problem $B$, symbolically $A \leq_m B$, if there exists a function $f$ from strings to strings that can be computed in polynomial time and that has the property that $w \in A$ if and only if $f(w) \in B$. Intuitively, an algorithm for problem $B$ can be used to solve problem $A$ with only polynomial overhead. In other words, $B$ must be at least as hard as $A$ with

respect to solvability in polynomial time.

## 2.3 NP-Hardness and NP-Completeness

A problem that has the property that all problems in NP can be polynomially reduced to it is obviously at least as hard as all problems in NP, i.e., it is NP-*hard* with respect to polynomial many-one reductions.

Since it is very difficult to prove something by quantifying over the entire class of languages in NP, it would be convenient to identify a problem $X$ that is in NP and NP-hard. Because of the transitivity of $\leq_m$ it then suffices to reduce $X$ to $Y$ in order to show NP-hardness of $Y$.

Problems that are NP-hard and in NP are called NP-*complete*. More generally, a problem is called $\mathcal{C}$-complete for a complexity class $\mathcal{C}$, if it is in $\mathcal{C}$ and $\mathcal{C}$-hard. Although it is not obvious that NP-complete problems exist, it turns out that a large number of natural problem for which we do not know polynomial algorithms are NP-complete.[3] NP-complete problems have the interesting property that a polynomial, deterministic algorithm for one of these problems implies that all problems in NP can be solved in polynomial times on a deterministic machine, i.e., it implies that P = NP. Since all attempts of finding polynomial algorithms for NP-complete problems have failed so far, the proof that a problem is NP-hard implies that no efficient algorithm is known for this problem according to the current state of the art. Further, because of the large number of unsuccessful attempts to find efficient methods for solving NP-complete problems it is nowadays *believed* that it is impossible to solve such problems in polynomial time.

The prototypical example of an NP-complete problem is SAT—the problem of determining whether a boolean formula $F$ over the set of boolean variables $V$ is satisfiable. SAT is obviously a problem in NP since we can guess a truth assignment for the variables in $V$ in $|V|$ steps and verify in polynomially many steps that this assignment makes $F$ true. Showing that SAT is also NP-hard is much more difficult. The proof is based on a *generic reduction* that assigns to each pair formed by a nondeterministic Turing machine $M$ and an instance $I$ a boolean formula $F$ that is satisfiable if, and only if, $I$ is accepted by $M$ in polynomial time.

The problem that is complementary to SAT, i.e., the problem to decide

[3] Garey and Johnson [1979] provide a list of approximately 300 NP-complete problems.

whether a boolean formula is unsatisfiable, is called UNSAT. Interestingly, UNSAT is not necessarily in NP, since no nondeterministic algorithm is known that accepts the corresponding language in polynomial time. UNSAT and all other problems complementary to problems in NP are assigned to the class coNP, and it is conjectured that NP $\neq$ coNP. Obviously, UNSAT is a coNP-complete problem under the definitions given above.

If one is only interested in showing that a problem is difficult, one often uses a different form of reduction, namely the *polynomial Turing reduction*. A problem $A$ can be reduced to $B$ by a polynomial Turing reduction, written $A \leq_T B$, if $A$ can be decided in polynomial time by a deterministic Turing machine $M$ that is allowed to use a procedure (a so-called *oracle*) for deciding $B$, whereby calling the procedure does only cost constant time. Under this type of reductions all problems that are NP-hard and coNP-hard with respect to polynomial many-one reductions are NP-hard. Further, a problem that is in NP and NP-hard under polynomial Turing reductions cannot be solved in polynomial time unless P = NP. If we use the term NP-hard in the following without any qualification, we mean (as usual) NP-hardness with respect to Turing reductions. This convention allows us to describe all difficult problems by the term "NP-hard."

### 2.4 Other Complexity Classes

In addition to P and NP, there exist a number of other so-called complexity classes.[4] On one hand, there are subclasses of P that are used to characterize problems that can be solved efficiently on parallel machines. This is a topic, however, that has not been extensively studied in AI (but cf. [Kasif, 1990; Kasif and Delcher, 1994]). On the other hand, there are classes that permit to characterize NP-hard problems more accurately, a topic we will discuss below.

## 3 Analyzing the Complexity of AI Problems

An essential prerequisite for a complexity analysis is a rigorous and unambiguous problem specification, something that one does not find very often in AI. Such a problem specification has the great advantage—independently

[4] Johnson [1990] gives a survey of the state of the art.

from a complexity analysis—that it can communicate the essential ideas of an approach. For instance, one of the first papers on the complexity of knowledge representation and reasoning [Brachman and Levesque, 1984] contains a concise and elegant specification of the semantics for so-called *description logics* (also called *terminological logics* or *concept languages*) [Brachman and Schmolze, 1985; Nebel and Smolka, 1991; Schmolze and Woods, 1992].

The observation that an AI problem is NP-hard, however, is usually not very interesting in itself since, as mentioned in the Introduction, most AI problems appear to be difficult in any case. Such an observation is interesting only if there are different opinions about the difficulty, as in the case of the blocks-world planning problem [Gupta and Nau, 1992], or if it is claimed that a problem can be solved efficiently [Nebel, 1988]. Further, an NP-hardness result alone does not provide many insights about the problem. It does not tell us which aspects of the problem are responsible for the difficulty, it does not tell us which aspects can nevertheless be solved efficiently and it gives no information about the relationship to other problems.

## 3.1 Polynomial Special Cases

One of the most interesting questions in the context of NP-hard problems is whether the problem specification is too general, and whether reasonable restrictions might lead to an efficiently solvable problem.

For instance, although SAT is an NP-complete problem, there are syntactically restricted variants that are efficiently solvable. For example, 2SAT, the satisfiability problem for boolean formulas in conjunctive normal form with only two literals per clause, can be solved in polynomial time.[5] Further, the problem of deciding whether a conjunction of Horn formulae is satisfiable, is a polynomial time problem.

Similar observations can be made in the area of knowledge representation formalisms, and often an application requires only the syntactically restricted form that allows efficient inferences. Let us consider as an example reasoning about *qualitative temporal information*.

Temporal information is often conveyed qualitatively by specifying the relative positions of time intervals such as "...point to the figure while ex-

[5] A straightforward argument proving this claim is based on the observation that resolution with clauses of length two leads only to resolvents of length two, of which there are only quadratically many.

| Basic Interval Relation | Sym-bol | Endpoint Relations |
|---|---|---|
| $X$ before $Y$ | $\prec$ | $X^- < Y^-, \quad X^- < Y^+,$ |
| $Y$ after $X$ | $\succ$ | $X^+ < Y^-, \quad X^+ < Y^+$ |
| $X$ meets $Y$ | m | $X^- < Y^-, \quad X^- < Y^+,$ |
| $Y$ met-by $X$ | $\mathsf{m}^\smile$ | $X^+ = Y^-, \quad X^+ < Y^+$ |
| $X$ overlaps $Y$ | o | $X^- < Y^-, \quad X^- < Y^+,$ |
| $Y$ overlapped-by $X$ | $\mathsf{o}^\smile$ | $X^+ > Y^-, \quad X^+ < Y^+$ |
| $X$ during $Y$ | d | $X^- > Y^-, \quad X^- < Y^+,$ |
| $Y$ includes $X$ | $\mathsf{d}^\smile$ | $X^+ > Y^-, \quad X^+ < Y^+$ |
| $X$ starts $Y$ | s | $X^- = Y^-, \quad X^- < Y^+,$ |
| $Y$ started-by $X$ | $\mathsf{s}^\smile$ | $X^+ > Y^-, \quad X^+ < Y^+$ |
| $X$ finishes $Y$ | f | $X^- > Y^-, \quad X^- < Y^+,$ |
| $Y$ finished-by $X$ | $\mathsf{f}^\smile$ | $X^+ > Y^-, \quad X^+ = Y^+$ |
| $X$ equals $Y$ | $\equiv$ | $X^- = Y^-, \quad X^- < Y^+,$ $X^+ > Y^-, \quad X^+ = Y^+$ |

Table 1: The set **B** of the thirteen basic relations.

plaining the performance of the system ..." Consequently, for natural language understanding [Allen, 1984; Song and Cohen, 1988], general planning [Allen, 1991; Allen and Koomen, 1983], and presentation planning in a multimedia context [Feiner *et al.*, 1993], the representation of qualitative temporal relations and reasoning about them is essential.

Allen [1983] introduces a calculus of binary relations on intervals for representing qualitative temporal information and addresses the problem of reasoning about such information. This approach to reasoning about time is based on the notion of *time intervals* and *binary relations* on them. A *time interval* $X$ is an ordered pair $(X^-, X^+)$ such that $X^- < X^+$, where $X^-$ and $X^+$ are interpreted as points on the real line.

Given two time intervals, their relative positions can be described by *exactly one* of the elements of the set **B** of thirteen *basic interval relations* (denoted by $B$ in the following), where each basic relation can be defined in terms of its *endpoint relations* (see Table 1).

In order to express indefinite information, unions of the basic interval relations are used, which are written as sets of basic relations leading to $2^{13}$

binary indefinite *interval relations*. The set of all binary interval relations $2^{\mathbf{B}}$ is denoted by $\mathcal{A}$.

An atomic formula of the form $X\{B_1, \ldots, B_n\}Y$ is called *interval formula*. Such a formula is said to be *satisfied* by an *assignment* of time intervals to $X$ and $Y$ if one of the basic relations $B_i$ holds between the two assigned intervals. A finite set of interval formulas $\Theta$ is *satisfiable* if there exists an assignment that satisfies all formulas in the set.

As an example consider three intervals $X_1, X_2, X_3$ such that $X_1$ is strictly inside $X_2$ and $X_3$ lies after $X_1$ but strictly inside $X_2$:

$$(X_1\{\mathsf{d}\}X_2) \quad (X_3\{\succ, \mathsf{m}^\smile\}X_1) \quad (X_3\{\mathsf{d}\}X_2)$$

This set is satisfiable, while the addition of $(X_3\{\mathsf{s}\}X_2)$ would make it unsatisfiable.

The fundamental *reasoning problem* in this framework is the problem of deciding the *satisfiability* of a set of interval formulae, also called ISAT in the following. This problem is fundamental because all other interesting problems can be polynomially reduced to it [Golumbic and Shamir, 1992]. Unfortunately however, ISAT is NP-hard, as has been shown by Vilain and Kautz [1986].

If one uses Allen's calculus in the context of natural language understanding for reasoning about the relative positions of events mentioned in a story, however, it turns out that only a subset of all possible relations is necessary [Song and Cohen, 1988]. This subset has the property that each relation can be described as a conjunction of arithmetic comparisons between the interval endpoints. For example, all basic relations belong to this subset, as well as the relation $\{\mathsf{d}, \mathsf{o}, \mathsf{s}\}$, which can be characterized by the conjunction

$$(X^- < X^+) \wedge (Y^- < Y^+) \wedge (X^- < Y^+) \wedge (Y^- < X^+) \wedge (X^+ < Y^+)$$

The same subset of relations is sufficient for diagnosis of technical artifacts [Nökel, 1991]. Interestingly, the reasoning problems for the restricted set of relations are efficiently solvable [Vilain *et al.*, 1989; van Beek and Cohen, 1990].

Another problem that is NP-hard in general but seems to be easily solvable in practice is the problem of interpreting "sketch maps" as they can be derived from aerial photographs [Reiter and Mackworth, 1989]. Interpretation means here to assign a meaning such as "shore," "river," "street," etc.

to the lines on the sketch map. As has been shown by Selman [1994], the problem is only NP-hard if we think that it is possible that the source of a river can be on a street. If such a configuration is impossible, however, the problem can be formalized as a 2SAT problem, and is efficiently solvable for this reason. Hence, if we exclude such unlikely interpretations, sketch map interpretation is an easy problem.

## 3.2 Classifying all Special Cases

While identifying relevant polynomial special cases of an NP-hard problem is interesting in itself and may help to solve the problem at hand, it only provides us with some indications of the sources of complexity involved in the general problem. An analysis of all special cases according to some problem parameter and a determination of the exact boundary between polynomial and NP-hard special cases is, however, much more interesting.[6] Further, such a classification can be used in order to check quickly whether the special case appearing in an application can be solved efficiently.

One example for such a complete classification is the paper by Donini *et al.* [1991a], in which the complexity of the basic reasoning problem for different *description logics* is analyzed. Description logics, which have their roots in the knowledge representation formalism KL-ONE [Brachman and Schmolze, 1985], have been developed to support the representation of the conceptual and terminological part of Artificial Intelligence applications. The main computational services provided by description logic systems are the computation of the *concept hierarchy* according to the *subsumption* relation between concepts and the computation of *instance relationships* between concepts and objects of the application domain.[7]

In order to describe concepts, we start with an alphabet $\mathbf{C}$ of *concept symbols* (denoted by $A$) and an alphabet $\mathbf{R}$ of *role symbols* (denoted by $P$), which are disjoint. Concept symbols are intended to denote some subset of a domain, and role symbols are intended to denote unary, set-valued functions or, equivalently, two-place relations on the domain. From concept and role

[6] One should note that it is in general impossible to identify a greatest polynomial subproblem of an NP-complete problem (provided $\mathsf{P} \neq \mathsf{NP}$) [Orponen *et al.*, 1986]. However, for a given problem parameter, the identification of the boundary may nevertheless be possible.

[7] See also the chapter on description logics in this volume.

symbols, complex *concept descriptions* (denoted by $C$) and complex *role descriptions* (denoted by $R$) are composed using a variety of description-forming operations.

In order to give an example, we will specify the language $\mathcal{FL}^-$ originally introduced by Brachman and Levesque [1984]:

$$C \ \rightarrow \ A \mid C \sqcap C' \mid \forall R.C \mid \exists R,$$

where roles can be specified only as atomic roles

$$R \ \rightarrow \ P.$$

Using this language, we can form various concept descriptions, such as "father" or "father all of whose children are professors:"

$$\mathit{Man} \sqcap \exists \mathit{child}$$
$$\mathit{Man} \sqcap \exists \mathit{child} \sqcap \forall \mathit{child}.\mathit{Professor}$$

The formal meaning of concept descriptions built according to the above rules is given by an interpretation $\mathcal{I} = (\mathbf{D}^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\mathbf{D}^{\mathcal{I}}$ (the *domain*) is an arbitrary, nonempty set and $\cdot^{\mathcal{I}}$ (the *interpretation function*) is a function such that:

$$\begin{aligned} A^{\mathcal{I}} &\subseteq \mathbf{D}^{\mathcal{I}} \\ P^{\mathcal{I}} &\subseteq \mathbf{D}^{\mathcal{I}} \times \mathbf{D}^{\mathcal{I}}. \end{aligned}$$

The denotation of complex concept descriptions is given inductively by:

$$\begin{aligned} (C \sqcap C')^{\mathcal{I}} &= C^{\mathcal{I}} \cap C'^{\mathcal{I}} \\ (\forall P\!:\! C)^{\mathcal{I}} &= \{d \in \mathbf{D}^{\mathcal{I}} \mid P^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\} \\ (\exists P)^{\mathcal{I}} &= \{d \in \mathbf{D}^{\mathcal{I}} \mid P^{\mathcal{I}}(d)\}. \end{aligned}$$

Based on this semantics, the notion of *subsumption* mentioned above is defined as set-inclusion. A concept $C$ is *subsumed* by another concept $C'$, written $C \preceq C'$, iff $(C)^{\mathcal{I}} \subseteq (C')^{\mathcal{I}}$ for every interpretation $\mathcal{I}$. For instance, we have

$$\mathit{Man} \sqcap \exists \mathit{child} \sqcap \forall \mathit{child}.\mathit{Professor} \ \preceq \ \mathit{Man} \sqcap \exists \mathit{child}$$

Based on this subsumption relation, a concept hierarchy can be computed. If the logic is extended to describe single objects by using role and concept

symbols, then the notion of *instance relationship* can be formalized as set-membership in concepts.

Note that one can think of quite different terminological logics employing, for instance, different role-forming operators, cardinality restrictions on roles, concept disjunction, concept negation and so on. Indeed, quite a number of different representation systems have been built using a variety of terminological logics (for a survey, see [Nebel, 1990a]).

Brachman and Levesque started with the description logic specified above and showed that the subsumption problem can be solved in polynomial time. The addition of an innocent looking role-forming operator

$$R \rightarrow P \mid (R|C)$$

with the following semantics

$$(R|C)^{\mathcal{I}} = \{(x,y) \in D \times D \mid (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$$

makes the subsumption problem NP-hard, however.[8] Brachman and Levesque [1984] argue that an exponential runtime behavior of a knowledge representation system using description logics is unacceptable and asked for the shape of the "computational cliff" they had identified.

It took seven years to come up with an exhaustive answer. Between 1987 and 1992, a number of results concerning complexity and decidability of subsumption for different description logics where published [Levesque and Brachman, 1987; Nebel, 1988; Patel-Schneider, 1989b; Schmidt-Schauß, 1989; Hollunder *et al.*, 1990; Schmidt-Schauß and Smolka, 1991; Donini *et al.*, 1991b; Schild, 1991; Donini *et al.*, 1992], but the boundary between polynomial and NP-hard description logic appeared still fuzzy. Finally, Donini *et al.* [1991a] were able to provide an exhaustive classification of all description logics that contain $\mathcal{FL}^-$ as a sublanguage.

Similar classifications were carried out in the area of plan generation. Bylander [1994] analyzed the computational complexity of propositional STRIPS planning [Fikes and Nilsson, 1971] according to different restrictions on operators and Bäckström and Nebel [1993] did a similar analysis for a slightly more expressive planning formalism taking local and global restrictions into account. In both cases, polynomiality is achieved only for very severe restrictions on the planning formalism, however.

[8] The proof appears in [Levesque and Brachman, 1987].

In both cases described above, it was possible to analyze the complexity of the subproblems manually. Sometimes, the space of subproblems may be so large, however, that a manual analysis is impossible. If one tries to analyze all possible subproblems in Allen's interval calculus according to the allowed relations, it turns out that there are $2^{2^{13}}$ subproblems. By analyzing the structure of the space of subproblems and using a computer-based case analysis, it was nevertheless possible to identify the precise boundary between polynomial and NP-hard subproblems also in this case [Nebel and Bürckert, 1994].

## 3.3 Differential Analyses

Often a reasoning problem is only informally specified and there exist different formalizations, which may lead to different complexity classes for the formalized reasoning problems. In this situation, the complexity results might be used as one criterion to choose between different formalization.

For instance, there exist a number of formalizations of *defeasible inheritance* in so-called *inheritance networks*. Inheritance networks are composed out of *nodes* denoting individuals, concepts, or properties, and *positive* and *negative links* between nodes standing for "$X$ is typically $Y$," symbolically $X \rightarrow Y$, and "$X$ is typically not a $Y$," symbolically $X \not\rightarrow Y$ [Fahlman, 1979; Touretzky, 1986]. A typical example of an inheritance network is given in Figure 1.

From the network in Figure 1 we may conclude, for instance, that *Clyde* is a member of the class of *royal elephants*, which are typically *elephants*, which are in turn typically *grey*. This would usually imply that it is very likely that *Clyde* is *grey* – if there is no information to the contrary. In our case, however, there is an explicit indication that *royal elephants* are typically not *grey*, which *overrides* the more general statement about the color of elephants.

Fahlman [1979] gives a formalization of the defeasible inheritance problem based on an efficient marker propagation method. As Touretzky [1986] points out, however, this method can lead to counter-intuitive results. For this reason, Touretzky provides a calculus for defeasible inheritance trying to circumvent the problems in Fahlman's approach. In Touretzky's approach *conclusions* are supported by *argument chains* built from links in the network. For instance, the conclusion "*Clyde* $\not\rightarrow$ *grey*" is supported by the argument

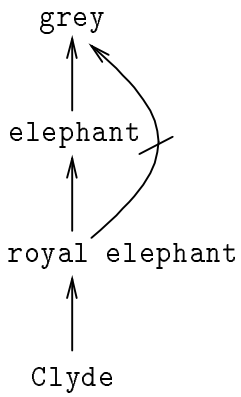


Figure 1: Example of an inheritance network

chain "*Clyde* $\rightarrow$ *royal elephant* $\not\rightarrow$ *grey*. Touretzky then specifies a method to derive sets of *coherent* argument chains meaning that

1. longer argument chains are created from shorter ones;
2. sets of argument chains are not allowed to lead to contradictory conclusions;
3. if there is more specific information, this information must be used.

Maximal sets of argument chains are also called *extensions*. Ignoring the technical details, it should be nevertheless obvious that in the general case one can derive more than one extension from a network. One way to cope with this problem is to accept all conclusions that are supported by at least one extension, called *credulous* reasoning. Another way to deal with the problem is to accept only those conclusions that are supported by all extensions, called *ideally skeptical* reasoning.

Although Touretzky's formalization of defeasible reasoning leads to more intuitive results than Fahlman's method, it is possible to have different opinions about whether Touretzky's proposal is the most reasonable one. In fact, there exist a number of different proposals for the definition of what an extension is [Touretzky *et al.*, 1987]. There exist alternative proposals

for the concatenation of argument chains, for conflict resolution, and for the notion of specificity. All in all, there are (at least) eight different ways of interpreting inheritance networks. Furthermore, it seems to be the case that it is not clear what the "best" interpretation is.

This leads to the interesting problem of identifying the interpretation with the best computational properties. Selman and Levesque [1989; 1993] have analyzed the following reasoning problems:

- Computation of the set of conclusions supported by one (arbitrary) extension.
- Deciding whether a conclusion follows by *ideally skeptical* or *credulous* reasoning.

Since an extension may contain exponentially many argument chains, it is computationally infeasible to compute a set of conclusions by generating the extension. As shown by Selman and Levesque [1993], one source of complexity is the way how argument chains are concatenated. In the case of Touretzky's *coupled* concatenation, it is NP-hard to generate a conclusion set. If one uses *decoupled* inheritance [Horty *et al.*, 1987], however, conclusion sets can be generated in polynomial time.

Unfortunately, this positive result does not carry over to ideally skeptical and credulous reasoning, because in this case *all* extensions must be considered—and there may be exponentially many extensions. Polynomiality can only be guaranteed if there is a limited number of extensions. This can be achieved by what is called *skeptical* inheritance (not to be confused with ideally skeptical reasoning). In this inheritance regime, an argument chain is only included in an extension if there is no *potentially valid* contradicting argument chain, and for this reason, this inheritance regime generates only one extension.

Summarizing, the result of Selman and Levesque's analysis is that the only way to guarantee efficiency in defeasible inheritance is the adoption of *skeptical, decoupled* inheritance. In particular, it also follows that Touretzky's [1986] method of "compiling" inheritance networks into structures that allow for applying Fahlman's efficient marker propagation method uses exponential time.

## 3.4 Using Complexity Analyses to Check for Plausibility

Sometimes it is claimed that there exist certain relationships between different problems or methods, for instance, that problem $P$ is a subproblem of another problem $Q$ or that a particular method $A$ is more efficient then a method $B$. Such claims are usually justified by intuitively plausible arguments. However, such claims should not only make sense from an intuitive point of view, but they should also be in accordance with the computational properties of the problems or methods, which is not always the case.

Dean and Boddy [1987; 1988], for instance, claim that the problem of *temporal projection* in partially ordered sets of event is one of the important subproblems in *partial-order* (or nonlinear) *planning*. Based on this claim, they analyze the complexity of the temporal projection problem, which turns out to be NP-hard, even under severe restrictions. For this reason, they specify an efficient approximation method that solves the projection problem only partially, but in polynomial time. However, from a computational point of view, the underlying claim seems to be arguable.

The *temporal projection* problem is the problem of determining for a given *initial state* and a given *partially ordered set of actions* the conditions that hold *necessarily* (i.e., in all linearizations) before a given event.[9] This problem is analyzed by Dean and Boddy in the context of propositional STRIPS planning [Fikes and Nilsson, 1971].

Formally, Dean and Boddy [1988] define a (propositional) state $S$ to be a set of propositional atoms. Events are sets of triples of the form

$$e = \{\langle p_1, a_1, d_1\rangle, \ldots, \langle p_n, a_n, d_n\rangle\},$$

with $p_i$, $a_i$ and $d_i$ being sets of propositional atoms standing for the (positive) *preconditions*, the *add-list* and the *delete-list*, respectively. An event containing only one triple is also called *unconditional event*. The occurrence of an event $e = \{\langle p_i, a_i, d_i\rangle\}$ in state $S$ results in a new state $S'$ that is computed by adding all $a_i$'s from $e$ to $S$ and deleting all $d_i$'s from this set for all triples $\langle p_i, a_i, d_i\rangle$ such that $p_i \subseteq S$. A sequence of events starting from an initial state $I$ leads to a final state $F$ by applying this rule iteratively.

[9] This problem can be varied by asking for the conditions that hold after an event or by asking for the conditions that hold *possibly*, i.e., in one linearization.

If we consider, for example, a set of two events $\{e_1, e_2\}$ that are specified as follows

$$\begin{aligned} e_1 &= \{\langle\{p\},\{q\},\{p\}\rangle\} \\ e_2 &= \{\langle\{p\},\{r\},\{s\}\langle\}, \end{aligned}$$

and which are unordered, we can ask whether $q$ holds necessarily after $e_1$ given the initial state $I = \{p\}$. This is indeed the case, since $e_1$ (in the sequence $e_1e_2$) results in the state $S = \{q\}$ and $e_2e_1$ results in the state $S' = \{r, q\}$. Similarly, one can easily verify that $r$ holds possibly, but not necessarily after $e_2$.

While it is easy to compute the consequences of a sequence of events on a given initial state, the problem of computing consequences for partially ordered event sets seems to be difficult. Since there are potentially exponentially many linearizations of the partial order, one may suspect that this problem is NP-hard. As mentioned above, Dean and Boddy [1988] showed that this indeed the case, even for severe restrictions, such as restricting the events to be unconditional.

In fact, it turns out that temporal projection is even harder than thought. Dean and Boddy conjectured that the problem becomes polynomial under the following restrictions:

1. only unconditional events,
2. the precondition set, the add list and the delete list are singletons and the precondition set is identical with the delete list, and
3. the initial state is a singleton set.

However, even in this case, temporal projection remains NP-hard [Nebel and Bäckström, 1994]. Interestingly, if we apply the same restrictions to the plan generation problem—which is an NP-hard problem in the general case—it becomes polynomial. For this reason, there is the question whether temporal projection is indeed a subproblem of planning. Dean and Boddy [1988] seem to assume that the temporal projection problem is in particular important for checking the *validity* of a plan, i.e., for checking whether the plan is executable and leads indeed to the desired goal.

A plan is executable if the preconditions of all actions are satisfied. If we are dealing with partially ordered sets of actions (or events), this must, of

course, hold for all linearizations. In the general case, temporal projection and plan validation have the same complexity and it seems to be the case that there is no natural decomposition of the validity problem into temporal projection problems. However, if we restrict ourselves to unconditional actions, there exists a quite natural decomposition. A partial-order plan consisting of only unconditional actions is valid if, and only if, the preconditions of all actions are necessarily satisfied and the goal specification is necessarily true after all actions have been executed. In other words, it is possible to reduce the validity problem to a sequence of temporal projection problems. While this sounds like a clever *divide and conquer* strategy, it turns out to be the opposite if one takes a computational point of view. Deciding the validity for partial-order plans consisting of unconditional actions can be shown to a be polynomial time problem [Nebel and Bäckström, 1994] using the technique developed by Chapman [Chapman, 1987]. Deciding the corresponding temporal projection problems is NP-hard, however. For this reason, Dean and Boddy's claim that temporal projection is a substantial subproblem of planning (or of plan validation) seems to be at least arguable.

Another area where computational complexity analysis has been applied to probe for the plausibility of some hypotheses is the area of *case-based planning*. While it seems intuitively plausible that it makes sense to reuse old solutions when generating solutions for a new planning situation, there is the question under which circumstances we might achieve a provable speed-up. Not very surprisingly, reusing old solutions never leads to a provable speed up in terms of computational complexity, even if we assume that the new situation differs only minimally from the old situation we already have a solution for [Nebel and Koehler, 1995]. In fact, if we take the suggestion from the literature seriously that the new solution should be generated by *minimally modifying* the old solution [Kambhampati and Hendler, 1992], there are cases when it is easier to generate a plan from scratch. Even worse, finding a solution in a case library that is appropriate to be used as the starting point for generating a new solution can already be computationally expensive [Nebel and Koehler, 1995].

While all these results concern "only" the worst-case complexity, empirical tests of different plan reuse techniques also seem to suggest that it is not obvious that plan reuse techniques pay off under all circumstances [Nebel and Koehler, 1995].

## 3.5 Classifying NP-hard Problems

Although the classification "polynomial vs. NP-hard" seems to be sufficient in most cases, it is sometimes worthwhile to determine the precise complexity of a problem. First of all, such a precise classification gives us a hint about how many sources of complexity we can expect to find, and, connected with that, it tells us how severe restrictions must be before we can expect to find polynomial subproblems. Secondly, a precise classification tells us something about the applicability of certain methods. If we can show, for instance, that a problem is NP-complete, it may be the case that methods for other NP-complete problems can be applied, while this is not the case for problems in higher complexity classes. Thirdly, a precise determination of the complexity can be used to compare the relative difficulty of different problems.

In Section 2, we remarked that a large number of NP-hard problems are also NP-complete or coNP-complete. However, there exist also NP-hard problems that do not seem to belong to NP or coNP. In order to classify those problems, other machine models and different resource restrictions are used. For instance, the class PSPACE is the class of decision problems that can be solved on deterministic Turing machines using polynomial space. Interestingly, the nondeterministic class NPSPACE (i.e., polynomial space on nondeterministic machines) is identical to PSPACE because a nondeterministic Turing machine can be simulated on a deterministic one with only quadratic space overhead. For this reason, we have obviously $\mathsf{NP} \subseteq \mathsf{PSPACE}$. As in the case of P and NP, it is unknown whether this inclusion is strict, but it is believed to be.

NP-complete as well as PSPACE-complete problems can be solved on deterministic Turing machines using exponential time. In other words, the theoretical difference between NP-complete and PSPACE-complete problems does not seem to be relevant for practical purposes. However, it turns out that PSPACE-complete problems require much more severe restrictions in order to achieve polynomiality. For instance, as mentioned above, plan generation in a propositional STRIPS framework appears to be a problem for which we have to make quite severe restrictions in order to achieve polynomiality. The deeper reason for this seems to be that the general problem is PSPACE-complete [Bylander, 1994]. Hence, the theoretical difference in complexity also seems to have practical consequences.

There exist also problems that appear slightly harder than NP-complete

problems but easier than the PSPACE-complete problems. One such problem is the problem of *belief revision*. This is the problem of changing a belief state, knowledge base, or database in order to accommodate new information that is possibly inconsistent with the belief state. This problem has been studied in philosophical logic [Alchourrón *et al.*, 1985], as well as in AI [Ginsberg, 1986] and the field of logical databases [Fagin *et al.*, 1983]. There are many different ways of revising a belief state rationally – according to the *rationality postulates* for revision that have been developed by Alchourrón, Makinson, and Gärdenfors [Alchourrón *et al.*, 1985]. In Computer Science, the preferred way seems to be to use a *syntax-based* revision scheme [Nebel, 1991a] that can be sketched as follows. Given a belief state $K$ represented by a finite set of propositions and a new proposition $\varphi$, the revised belief state, symbolically $K \dot{+} \varphi$, is the disjunction of all maximal subsets of $K$ consistent with $\varphi$ plus $\varphi$ itself.

Assume, for instance, that our belief state is represented by

$$K \quad = \quad \{p, r, p \rightarrow q\}.$$

Assume further that we want to revise $K$ by $\neg q$. The following two subsets of $K$ are maximal subsets consistent with $\neg q$:

$$\begin{aligned} K_1 &= \{r, p \rightarrow q\} \\ K_2 &= \{r, p\} \end{aligned}$$

Hence, the revised belief state could be represented by

$$K \dot{+} \neg q \quad = \quad \{r, \neg q\}.$$

The main *reasoning problem* in this context is to find out whether a proposition $\psi$ follows from $K \dot{+} \varphi$. Assuming propositional logic as the base logic, this problem is obviously coNP-hard, because $p \wedge \neg p$ follows from $\emptyset \dot{+} \varphi$ if and only if $\varphi$ is unsatisfiable. It seems very unlikely that the problem is complete for coNP, however, since one can also show that it is NP-hard (with respect to polynomial many-one reductions). Further, although the problem can be easily shown to be solvable in polynomial space, it seems impossible to show that is complete for PSPACE. In fact, the problem turns out to be complete for a complexity class that lies between NP and PSPACE, namely the class $\mathsf{coNP}^{\mathsf{NP}}$ (also denoted by $\Pi_2^p$) [Nebel, 1991a; Eiter and Gottlob,

1992]. This is the class of problems that are complementary to problems that can be solved on a nondeterministic Turing machine in polynomial time under the provision that it is allowed to access a so-called NP-oracle, i.e., a procedure that can solve an NP problem in constant time.

Intuitively, this means that the problem contains two sources of complexity. In fact, in the case of belief revision these two sources can be easily identified. Firstly, there is the classical logical reasoning problem, which is coNP-complete in case of propositional logic. Secondly, there is the problem of quantifying over the potentially exponentially many maximal consistent subsets of the belief state. This implies that in order to achieve polynomiality, both sources of complexity have to be eliminated. It is not enough to restrict the base logic to, say, propositional Horn logic [Eiter and Gottlob, 1992], but one also has to restrict the number of maximal, consistent subsets [Nebel, 1994].

Interestingly, a number of related reasoning problems can be classified as belonging to the same complexity class. For instance the *cautious* reasoning problems in *nonmonotonic logics*, such as default logic, autoepistemic logic, and circumscription are $\mathsf{coNP}^{\mathsf{NP}}$-complete [Gottlob, 1992; Stillman, 1992; Eiter and Gottlob, 1993b], as well as a number of *abduction problems* [Eiter and Gottlob, 1993a].[10] This implies that all these problems are equivalent on an abstract algorithmic level and that all these problems can be more or less directly translated into each other (e.g. [Gottlob, 1994]).

Furthermore, these results also enable us to check the plausibility of certain conjectures. Ben-Eliyahu and Dechter [1991] conjectured, for example, that their (polynomial-time) method of translating certain restricted default theories into classical propositional logic could also be applied to general default theories. As pointed out by Gottlob [Gottlob, 1992], this seems rather unlikely since it would mean that there exist a method of reducing a $\mathsf{coNP}^{\mathsf{NP}}$-complete problem to a coNP-complete problem implying that the two classes are identical, which is believed to be quite unlikely [Johnson, 1990].

[10] Cadoli and Schaerf [1993] gave an extensive survey of complexity results for nonmonotonic propositional logics.

# 4 Coping with NP-hardness

Having shown that a problem is NP-hard does not imply that it is impossible to tackle this problem computationally. It only means that we cannot hope to solve arbitrarily large instances of the problem in its full generality in reasonable time. However, this leaves us with a large number of options. We may concentrate on restricted version of the problem, we may go for approximate answers, we may use randomized algorithms or we may design exponential time algorithms that are practical for small to moderately sized instances.

In fact, coping with NP-hard problems can be viewed as one of the central themes of AI. Usually, in AI the complexity problem is tackled using a practical, experimental approach [Bylander, 1991] and theoretical considerations come second. Analytical, complexity theoretic analyses can then "only" explain why a certain approach is successful or why another one is likely to fail. However, such analyses are very important when the range of applicability of a certain technique is to be determined or when "scaling up" is an issue.

## 4.1 Natural Restrictions of a Problem

As has been noted in the previous section, often it is not necessary to solve a problem in its full generality, but it is sufficient to consider only a restricted form of the problem, which may turn out to be efficiently solvable. The examples considered were of the kind that the restriction was expressed in the form that only certain combinations of the basic vocabulary are allowed. Often, however, it may be some numerical parameter of the problem instance that can be blamed for the combinatorial explosion. In this case, it may turn out that (almost) all instances occurring in practice have this parameter bounded by a constant.

As an example, let us consider the two-level morphology that has been used for word-form recognition in natural language processing [Koskenniemi, 1983]. Barton [Barton, 1986] showed that this approach is NP-complete in the number of morphological rules, which means that the processing time can grow exponentially with the number of rules. A closer look reveals, however, that only a certain kind of rules, rules that realize co-called *harmony processes*, are responsible for the NP-completeness result. Furthermore, for all known languages it happens that only a very limited number of these

rules is necessary, namely at most two [Koskenniemi and Church, 1988]. In other words, for all practical cases, word-form recognition using two-level morphology is computationally feasible.

As a second example, we will consider subsumption checking in description logics. In most papers on the complexity of this problem, it is assumed that the interpretation of atomic concepts is unconstrained. However, in all existing knowledge representation systems that use description logics it is possible to *define* the meaning of an atomic concept [Heinsohn *et al.*, 1994]. For instance, using the description-forming operators introduced in Section 3.2, we may define the meaning of a *Father* and a *Proud-father* as follows:

$$\begin{aligned} \mathit{Father} &\doteq \mathit{Man} \sqcap \exists \mathit{child} \\ \mathit{Proud\text{-}father} &\doteq \mathit{Father} \sqcap \forall \mathit{child}.\mathit{Professor}. \end{aligned}$$

The formal interpretation of such defining equations is that the set of interpretations is restricted to those satisfying the equations, and subsumption is computed with respect to these satisfying interpretations.

In most cases, it is required that a set of such equations meets the following restrictions:

- every atomic concept appears at most once on the left hand side, and
- the definitions must be cycle-free, i.e., non-recursive,[11]

which means that one can view these equations as "macro definitions." Based on this observation, it is easily possible to reduce subsumption checking relative to a set of concept definitions to subsumption checking between two complex concept expressions that contain only undefined atomic concepts. One only has to replace each defined atomic concept by its definition.

While this reduction appears to be conceptually easy, it turns out that it may imply considerable computational costs. In the worst case, it can happen that the expansion process leads to an exponentially larger concept description [Nebel, 1990b]. Furthermore, this is not an artifact of the algorithmic technique, but the problem of subsumption checking relative to a set of equations seems to be inherently difficult. While subsumption checking in

[11] See [Baader, 1990; Nebel, 1991b] for an analysis of cyclic concept definitions.

$\mathcal{FL}^-$ is polynomial, as noted in Section 3.2, it becomes NP-hard if subsumption checking has to be done relative to a set of equations [Nebel, 1990b].

Interestingly, in all practically occurring situations the worst case never seems to happen. The reason is that definitions occurring in practice are somehow well-structured. They always have only limited depth which leads to a reasonable runtime behavior [Nebel, 1990b]; and this does not only hold for knowledge representation systems based on description logic but also for object-oriented database systems [Bergamaschi and Nebel, 1994].

In both cases described above, we have the situation that a problem that has been classified as difficult is nevertheless efficiently solvable in practice. Although the NP-hardness results seem to be irrelevant in these cases, they give us hints that the runtime requirement can be very sensitive with respect to the identified problem parameter, i.e., the number of harmony rules or the depth of definitions. In fact, an empirical analysis of different description logic systems [Heinsohn *et al.*, 1994] confirmed that the structure of the knowledge base can have indeed a severe influence on the runtime behavior.

While the restrictions on the above identified parameters lead provably to polynomiality, it can also be the case that a problem parameter predicts the difficulty of a problem with high probability. Empirical investigations of the graph coloring problem [Cheeseman *et al.*, 1991] and of the satisfiability problem for constant clause length [Mitchell *et al.*, 1992] seem to suggest that certain problem parameters determine easy and hard problem ranges. Basically, for one range of parameter values the problem is underconstrained, in which case it is easily solvable by almost all methods. For another range, the problem is overconstrained, leading to the same observation. Only in the range where approximately 50% of the instances lead to a "yes" answer, the problem appears to be difficult. These results, however, depend on certain assumptions concerning the distribution of problem instances leading to quite differing results. Gent and Walsh [1994], for instance, located the area of hard problems at another point than Mitchell et al [1992] because they used another distribution.

## 4.2 Enforced Restrictions

In an application-independent system one cannot simply assume that everything works out well. At least, one should be able to specify the range of applicability.

Brachman and Levesque [1984] go even one step further. They postulate that a knowledge representation system should have a status similar to a database system in conventional Computer Science applications. It should give a correct answer in a time bound that is predictable and reasonable [Levesque, 1986]. Based on their observation that a particular description-forming object can make subsumption NP-hard, they propose to restrict the expressiveness of representation languages as one way to achieve guaranteed efficiency.

This proposal, which does not seem to be unreasonable from a technical point of view, led to severe criticism, however [Doyle and Patil, 1991]. The main point of this criticisms was that restricting the representation language leads in most cases to the fact that the representation language becomes useless. While this is true in the general case, one should consider two points. First of all, Brachman and Levesque proposed this way of achieving efficiency only as one way. Secondly, there are indeed applications that can benefit from restricted knowledge representation languages [Wright *et al.*, 1993].

In any case, the discussion of this issue demonstrates that the idea of developing and analyzing knowledge representation systems independently from applications seems to be arguable. Applications pose varying requirements on the expressiveness and on the quality of replies, but efficiency almost always plays a prominent role.

## 4.3 Approximations

If it is impossible to identify restrictions that are tolerable, it may be possible to reduce the requirements on the answer quality. In particular, it is often possible to give up on the optimality of solutions. For example, generating arbitrary plans in the blocks world is polynomial while requiring optimal plans leads to NP-hardness [Gupta and Nau, 1992].

However, often AI problems cannot be weakened by giving up on optimality. In particular logic-based problems such as subsumption checking, diagnosis, or temporal projection are not optimization problems and it seems to be difficult to define what an approximation to a solution is.

One common solution is in this case to retreat to correct but incomplete methods. This means that the method only draws logically valid conclusions but may miss some. In order to be able to characterize the behavior of such methods formally, one can try to capture the incompleteness declaratively

using a weaker semantics. For example, Patel-Schneider [1989a] uses a four-valued semantics for characterizing a subsumption relation that is polynomial but weaker than the standard one. Levesque and Lakemeyer use a similar technique to weaken the general logical implication for propositional belief logics [Levesque, 1984; Lakemeyer, 1987].

Other approaches try to approximate the right answer from above and below by using correct but incomplete and incorrect but complete methods. For example, in the work by Cadoli and Schaerf [1991; 1992] sequences of interpretations are used to approximate the right answer from above and below. In Selman and Kautz' [1991] Horn theories are used to approximate arbitrary propositional theories from above and below. In this approach, it is assumed, however, that the approximating theories are computed "off-line" since their computation can be quite involved. First experiments using this approach show that this might be indeed an interesting way to go [Kautz and Selman, 1994].

While the above approaches provide us with incompleteness in a principled way that can be characterized declaratively, there is also a way of approximating the solution of a problem by methods that seem to work empirically well—without giving any guarantees at all. One such approach is the "greedy satisfiability" method, also called GSAT [Selman *et al.*, 1992], that tries to solve the satisfiability problem by local search attempting to maximize the number of satisfied clauses of a formula in conjunctive normal form. Local search is a well-known technique to solve optimization problems. However, it applicability to decision problems such as SAT is not *a priori* obvious. Nevertheless, GSAT works surprisingly well for a number of problem classes on which conventional techniques based on systematic search fail.

## 4.4 Reconsidering the Problem

If everything else fails, one may reconsider the problem and take an NP-hardness result as the starting point for radically redefining the problem. In particular, if the original problem formulation was intended to capture a cognitive capability which humans apply effortlessly, an NP-hardness result can be interpreted as stating that the problem formulation does not match the capability.

Enzinger et al [1994], for instance, consider a number of AI problems, which are formulated as NP-hard search problems, and contrast them with

simplifications and reformulations based on arguments from cognitive science. For example, they compare *case-based* diagnosis (see [Riesbeck and Schank, 1989]) with *consistency-based* diagnosis as formulated by Reiter [1987] and conclude that the former is much better behaved from a computational complexity point of view.

However, one should note that the application areas of the two approaches are very different. Case-based diagnosis works only if there exists experience with the technical device one wants to diagnose, which is not necessary for the consistency based approach. In other words, if one is interested in a diagnosis that uses only the knowledge about the structure and the functionality of a technical device, case-based analysis did not help at all to solve the problem at hand.

### 4.5 Solving Small Instances

The final way then is to accept the combinatorial explosion and to find efficient ways that help to solve instances as large as possible. Heuristics and structuring the search space in clever ways can help to do so. However, because of the inherent complexity, the instance size will always be limited to be only moderately large and there is no way to scale up to instances that are much larger (say, by a small constant factor).

## 5 Summary

If we view AI as the field concerned with the analysis and synthesis of cognitive capabilities using computation, then it appears to be natural to apply the tools that have been developed in theoretical computer science. In particular complexity theory is useful for analyzing the algorithmic structure of AI problems. Complexity analyses can help to identify sources of complexity, and to isolate aspects of a problem that may be efficiently solvable. Further, complexity theory can be used to compare and contrast different problems or problem formulations, and may be used to test hypotheses concerning the computational nature of problems.

In this context, an NP-hardness proof does not constitute the end of the analysis, but it marks the starting point for searching for efficient means to solve interesting aspects of the problems by whatever methods are available.

This may be done by restricting the problem, by developing approximation methods, or by radically redefining the problem. In a large number of cases, however, one must accept that exponential time is inevitable and the only question is how to solve instances up to some bounded size in reasonable time.

## Acknowledgements

I would like to thank Gerhard Brewka, Thomas Eiter, Georg Gottlob, Andrea Hemprich, Jana Koehler, Werner Nutt, and Uwe Schöning for comments on earlier versions of this paper.

## References

[Alchourrón *et al.*, 1985] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2):510–530, June 1985.

[Allen and Koomen, 1983] James F. Allen and Johannes A. Koomen. Planning using a temporal world model. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 741–747, Karlsruhe, Germany, August 1983.

[Allen, 1983] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[Allen, 1984] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[Allen, 1991] James F. Allen. Temporal reasoning and planning. In James F. Allen, Henry A. Kautz, Richard N. Pelavin, and Josh D. Tenenberg, editors, *Reasoning about Plans*, chapter 1, pages 1–67. Morgan Kaufmann, San Mateo, CA, 1991.

[Baader, 1990] Franz Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proceedings of the 8th National Conference of the American Association for Artificial Intelligence*, pages 621–626, Boston, MA, August 1990. MIT Press.

[Bäckström and Nebel, 1993] Christer Bäckström and Bernhard Nebel. Complexity results for $SAS^+$ planning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1430–1435, Chambery, France, August 1993. A long version of this paper appeared as Research Report LiTH-IDA-R-93-34, Computer and Information Science, Linköping University, Sweden.

[Balcázar *et al.*, 1988] José Luis Balcázar, Josep Diaz, and Joaquim Gabarró. *Structural Complexity I.* Springer-Verlag, Berlin, Heidelberg, New York, 1988.

[Balcázar *et al.*, 1990] José Luis Balcázar, Josep Diaz, and Joaquim Gabarró. *Structural Complexity II.* Springer-Verlag, Berlin, Heidelberg, New York, 1990.

[Barton, 1986] G. E. Barton. Computational complexity in two-level morphology. In *Proceedings of the 14th Annual Meeting of the ACL*, pages 53–59, New York, NY, 1986.

[Ben-Eliyahu and Dechter, 1991] Rachel Ben-Eliyahu and Rina Dechter. Default logic, propositional logic and constraints. In *Proceedings of the 9th National Conference of the American Association for Artificial Intelligence*, pages 379–385, Anaheim, CA, July 1991. MIT Press.

[Bergamaschi and Nebel, 1994] Sonia Bergamaschi and Bernhard Nebel. Automatic building and validation of complex object database schemata supporting multiple inheritance. *Applied Intelligence*, 4(2):185–204, May 1994.

[Brachman and Levesque, 1984] Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings of the 4th National Conference of the American Association for Artificial Intelligence*, pages 34–37, Austin, TX, 1984.

[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, April 1985.

[Bylander, 1991] Tom Bylander. Tractability and Artificial Intelligence. *Journal of Experimental and Theoretical Artificial Intelligence*, 3:171–178, 1991.

[Bylander, 1994] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1–2):165–204, 1994.

[Cadoli and Schaerf, 1991] Marco Cadoli and Marco Schaerf. Approximate entailment. In E. Ardizzone, S. Gaglio, and F. Sorbello, editors, *Trends in Artificial Intelligence: Proc. of the 2nd Congress of the Italian Association for Artificial Intelligence, AI*IA*, pages 68–77. Springer-Verlag, Berlin, Heidelberg, New York, 1991.

[Cadoli and Schaerf, 1992] Marco Cadoli and Marco Schaerf. Approximation in concept description languages. In B. Nebel, W. Swartout, and C. Rich, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 3rd International Conference*, pages 342–353, Cambridge, MA, October 1992. Morgan Kaufmann.

[Cadoli and Schaerf, 1993] Marco Cadoli and Marco Schaerf. A survey of complexity results for non-monotonic logics, 1993.

[Chapman, 1987] David Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32(3):333–377, July 1987.

[Cheeseman *et al.*, 1991] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the *really* hard problems are. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 331–337, Sydney, Australia, August 1991. Morgan Kaufmann.

[Dean and Boddy, 1987] Thomas L. Dean and Mark Boddy. Incremental causal reasoning. In *Proceedings of the 6th National Conference of the American Association for Artificial Intelligence*, pages 196–201, Seattle, WA, July 1987.

[Dean and Boddy, 1988] Thomas L. Dean and Mark Boddy. Reasoning about partially ordered events. *Artificial Intelligence*, 36(3):375–400, October 1988.

[Donini *et al.*, 1991a] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. Tractable concept languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 458–465, Sydney, Australia, August 1991. Morgan Kaufmann.

[Donini *et al.*, 1991b] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept languages. In J. A. Allen, R. Fikes, and E. Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference*, pages 151–162, Cambridge, MA, April 1991. Morgan Kaufmann.

[Donini *et al.*, 1992] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Bernhard Hollunder, Werner Nutt, and Alberto Marchetti Spacamella. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 53(2-3):309–327, 1992.

[Doyle and Patil, 1991] Jon Doyle and Ramesh S. Patil. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence*, 48(3):261–298, April 1991.

[Eiter and Gottlob, 1992] Thomas Eiter and Georg Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence*, 57:227–270, 1992.

[Eiter and Gottlob, 1993a] Thomas Eiter and Georg Gottlob. The complexity of logic-based abduction. In P. Enjalbert, A. Finkel, and K. W. Wagner, editors, *Proceedings Tenth Symposium on Theoretical Aspects of Computing STACS-93*, pages 70–79, Würzburg, Germany, February 1993. Springer-Verlag. Extended version to appear in JACM.

[Eiter and Gottlob, 1993b] Thomas Eiter and Georg Gottlob. Propositional circumscription and extended closed world reasoning are $\pi_2^p$-complete. *Theoretical Computer Science*, 114(2):231–245, 1993. Addendum 118:315.

[Enzinger *et al.*, 1994] Andrea Enzinger, Frank Puppe, and Gerhard Strube. Problemlösen ohne Suchen? *KI*, 1/94:73–81, 1994.

[Fagin *et al.*, 1983] Ronald Fagin, Jeffrey D. Ullman, and Moshe Y. Vardi. On the semantics of updates in databases. In *2nd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 352–365, Atlanta, Ga., 1983.

[Fahlman, 1979] Scott E. Fahlman. *A System for Representing and Using Real-World Knowledge*. MIT Press, Cambridge, MA, 1979.

[Feiner *et al.*, 1993] Steven K. Feiner, Diane J. Litman, Kathleen R. McKeown, and Rebecca J. Passonneau. Towards coordinated temporal multimedia presentation. In M. Maybury, editor, *Intelligent Multi Media*. AAAI Press, Menlo Park, CA, 1993. Forthcoming.

[Fikes and Nilsson, 1971] Richard E. Fikes and Nils Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability—A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.

[Gent and Walsh, 1994] Ian P. Gent and Toby Walsh. The hardest random sat problems. In B. Nebel and L. Dreschler-Fischer, editors, *KI-94: Advances in Artificial Intelligence*, pages 355–366, Saarücken, Germany, 1994. Springer-Verlag.

[Ginsberg, 1986] Matthew L. Ginsberg. Counterfactuals. *Artificial Intelligence*, 30(1):35–79, October 1986.

[Golumbic and Shamir, 1992] Martin C. Golumbic and Ron Shamir. Algorithms and complexity for reasoning about time. In *Proceedings of the 10th National Conference of the American Association for Artificial Intelligence*, pages 741–747. MIT Press, San Jose, CA, July 1992. A long version will appear in JACM.

[Gottlob, 1992] Georg Gottlob. Complexity results for nonmonotonic logics. *Journal for Logic and Computation*, 2(3), 1992.

[Gottlob, 1994] Georg Gottlob. The power of beliefs or translating default logic into standard autoepistemic logic. In Gerhard Lakemeyer and Bernhard Nebel, editors, *Foundations of Knowledge Representation*, volume 810 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, Heidelberg, New York, 1994. Also appeared in Proc. 13th IJCAI.

[Gupta and Nau, 1992] Naresh Gupta and Dana S. Nau. On the complexity of blocks-world planning. *Artificial Intelligence*, 56(2):223–254, 1992.

[Heinsohn *et al.*, 1994] Jochen Heinsohn, Daniel Kudenko, Bernhard Nebel, and Hans-Jürgen Profitlich. An empirical analysis of terminological representation systems. *Artificial Intelligence*, 68(2):367–397, August 1994.

[Hollunder *et al.*, 1990] Bernhard Hollunder, Werner Nutt, and Manfred Schmidt-Schauß. Subsumption algorithms for concept description languages. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 348–353, Stockholm, Sweden, August 1990. Pitman.

[Horty *et al.*, 1987] John F. Horty, Richmond H. Thomason, and David S. Touretzky. A skeptical theory of inheritance in nonmonotonic semantic networks. In *Proceedings of the 6th National Conference of the American Association for Artificial Intelligence*, pages 358–363, Seattle, WA, July 1987.

[Johnson, 1990] David S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. A*, pages 67–161. MIT Press, 1990.

[Kambhampati and Hendler, 1992] Subbarao Kambhampati and James A. Hendler. A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence*, 55:193–258, 1992.

[Kasif and Delcher, 1994] Simon Kasif and Arthur L. Delcher. Local consistency in parallel constraint networks. *Artificial Intelligence*, 69(1–2):307–327, 1994.

[Kasif, 1990] Simon Kasif. On the parallel complexity of discrete relaxation in constraint networks. *Artificial Intelligence*, 45(3):275–286, October 1990.

[Kautz and Selman, 1994] Henry Kautz and Bart Selman. An empirical evaluation of knowledge compilation. In *Proceedings of the 12th National Conference of the American Association for Artificial Intelligence*, pages 155–161, Seattle, WA, July 1994. MIT Press.

[Koskenniemi and Church, 1988] Kimmo Koskenniemi and Kenneth Ward Church. Complexity, two-level morphology and Finish. In *Proceedings of the 12th International Conference on Computational Linguistics*, pages 335–340, Budapest, Hungary, 1988.

[Koskenniemi, 1983] Kimmo Koskenniemi. Two-level morphology: A general computational model for word-form recognition and production. Technical Report No. 11, University of Helsinki, Department of General Linguistics, Helsinki, Finland, 1983.

[Lakemeyer, 1987] G. Lakemeyer. Tractable meta-reasoning in propositional logics of belief. In J. McDermott, editor, *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 402–408, Milan, Italy, August 1987. Morgan Kaufmann.

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

[Levesque, 1984] Hector J. Levesque. A logic of implicit and explicit belief. In *Proceedings of the 4th National Conference of the American Association for Artificial Intelligence*, pages 198–202, Austin, TX, 1984.

[Levesque, 1986] Hector J. Levesque. Making believers out of computers. *Artificial Intelligence*, 30(1):81–108, October 1986.

[Mitchell *et al.*, 1992] David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distributions of SAT problems. In *Proceedings of the 10th National Conference of the American Association for Artificial Intelligence*, pages 459–465, San Jose, CA, July 1992. MIT Press.

[Nebel and Bäckström, 1994] Bernhard Nebel and Christer Bäckström. On the computational complexity of temporal projection, planning, and plan validation. *Artificial Intelligence*, 66(1):125–160, 1994.

[Nebel and Bürckert, 1994] Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. In *Proceedings of the 12th National Conference of the American Association for Artificial Intelligence*, pages 356–361, Seattle, WA, July 1994. MIT Press. Extended version to appear in JACM.

[Nebel and Koehler, 1995] Bernhard Nebel and Jana Koehler. Plan reuse versus plan generation: A theoretical and empirical analysis. *Artificial Intelligence*, 1995. To appear. A preliminary version is available as DFKI Research Report RR-93-33.

[Nebel and Smolka, 1991] Bernhard Nebel and Gert Smolka. Attributive description formalisms ... and the rest of the world. In Otthein Herzog and Claus-Rainer Rollinger, editors, *Text Understanding in LILOG*, volume 546 of *Lecture Notes in Artificial Intelligence*, pages 439–452. Springer-Verlag, Berlin, Heidelberg, New York, 1991.

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

[Nebel, 1990a] Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, Heidelberg, New York, 1990.

[Nebel, 1990b] Bernhard Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.

[Nebel, 1991a] Bernhard Nebel. Belief revision and default reasoning: Syntax-based approaches. In J. A. Allen, R. Fikes, and E. Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference*, pages 417–428, Cambridge, MA, April 1991. Morgan Kaufmann.

[Nebel, 1991b] Bernhard Nebel. Terminological cycles: Semantics and computational properties. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 331–362. Morgan Kaufmann, San Mateo, CA, 1991.

[Nebel, 1994] Bernhard Nebel. Base revision operations and schemes: Semantics, representation, and complexity. In *Proceedings of the 11th European Conference on Artificial Intelligence*, pages 341–345, Amsterdam, The Netherlands, August 1994. Wiley.

[Nökel, 1991] Klaus Nökel. *Temporally Distributed Symptoms in Technical Diagnosis*, volume 517 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, Heidelberg, New York, 1991.

[Orponen *et al.*, 1986] P. Orponen, D. A. Russo, and U. Schöning. Optimal approximations and polynomially levelable sets. *SIAM Journal on Computing*, 15(2):399–408, 1986.

[Papadimitriou, 1994] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[Patel-Schneider, 1989a] Peter F. Patel-Schneider. A four-valued semantics for terminological logics. *Artificial Intelligence*, 38(3):319–351, April 1989.

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

[Reiter and Mackworth, 1989] Ray Reiter and Alan K. Mackworth. A logical framework for depiction and image interpretation. *Artificial Intelligence*, 41:125–155, 1989.

[Reiter, 1987] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, April 1987.

[Rich and Knight, 1991] Elaine Rich and Kevin Knight. *Artificial Intelligence*. McGraw-Hill, New York, NY, 2nd edition edition, 1991.

[Riesbeck and Schank, 1989] Christopher K. Riesbeck and Roger C. Schank. *Inside Case-Based Reasoning*. Erlbaum, Hillsdale, NJ, 1989.

[Schild, 1991] Klaus 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, August 1991. Morgan Kaufmann.

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

[Schmidt-Schauß, 1989] Manfred Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In R. Brachman, H. J. Levesque, and R. Reiter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 1st International Conference*, pages 421–431, Toronto, ON, May 1989. Morgan Kaufmann.

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

[Selman and Kautz, 1991] Bart Selman and Henry Kautz. Knowledge compilation using Horn approximations. In *Proceedings of the 9th National Conference of the American Association for Artificial Intelligence*, pages 904–909, Anaheim, CA, July 1991. MIT Press.

[Selman and Levesque, 1989] Bart Selman and Hector J. Levesque. The tractability of path-based inheritance. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, MI, August 1989. Morgan Kaufmann.

[Selman and Levesque, 1993] Bart Selman and Hector J. Levesque. The complexity of path-based defeasible inheritance. *Artificial Intelligence*, 62:303–339, 1993.

[Selman *et al.*, 1992] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the 10th National Conference of the American Association for Artificial Intelligence*, pages 440–446, San Jose, CA, July 1992. MIT Press.

[Selman, 1994] Bart Selman. Domain-specific complexity tradeoffs. In *Proceedings of the 11th European Conference on Artificial Intelligence*, pages 416–420, Amsterdam, The Netherlands, August 1994. Wiley.

[Song and Cohen, 1988] Fei Song and Robin Cohen. The interpretation of temporal relations in narrative. In *Proceedings of the 7th National Conference of the American Association for Artificial Intelligence*, pages 745–750, Saint Paul, MI, August 1988.

[Stillman, 1992] J. Stillman. The complexity of propositional default logics. In *Proceedings of the 10th National Conference of the American Association for Artificial Intelligence*, pages 794–799, San Jose, CA, July 1992. MIT Press.

[Touretzky *et al.*, 1987] David S. Touretzky, John F. Horty, and Richmond H. Thomason. A clash of intuitions: The current state of nonmonotonic multiple inheritance systems. In J. McDermott, editor, *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 476–482, Milan, Italy, August 1987. Morgan Kaufmann.

[Touretzky, 1986] David S. Touretzky. *The Mathematics of Inheritance Systems*. Morgan Kaufmann, Los Altos, CA, 1986.

[van Beek and Cohen, 1990] Peter van Beek and Robin Cohen. Exact and approximate reasoning about temporal relations. *Computational Intelligence*, 6:132–144, 1990.

[Vilain and Kautz, 1986] Marc B. Vilain and Henry A. Kautz. Constraint propagation algorithms for temporal reasoning. In *Proceedings of the 5th National Conference of the American Association for Artificial Intelligence*, pages 377–382, Philadelphia, PA, August 1986.

[Vilain *et al.*, 1989] Marc B. Vilain, Henry A. Kautz, and Peter G. van Beek. Constraint propagation algorithms for temporal reasoning: A revised report. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufmann, San Mateo, CA, 1989.

[Wright *et al.*, 1993] Jon R. Wright, Elia S. Weixelbaum, Gregg T. Vesonder, Karen E. Brown, Stephen R. Palmer, Jay I. Berman, and Harry H. Moore. A knowledge-based configurator that supports sales, engineering, and manufacturing at AT&T network systems. *The AI Magazine*, 14(3):69–80, 1993.