Knowledge Compilation Using Horn Approximations

Bart Selman and Henry Kautz
AI Principles Research Department
AT&T Bell Laboratories
Murray Hill, NJ 07974
{selman, kautz}@research.att.com

Abstract

We present a new approach to developing fast and efficient knowledge representation systems. Previous approaches to the problem of tractable inference have used restricted languages or incomplete inference mechanisms — problems include lack of expressive power, lack of inferential power, and/or lack of a formal characterization of what can and cannot be inferred. To overcome these disadvantages, we introduce a *knowledge compilation* method. We allow the user to enter statements in a general, unrestricted representation language, which the system compiles into a restricted language that allows for efficient inference. Since an exact translation into a tractable form is often impossible, the system searches for the best approximation of the original information. We will describe how the approximation can be used to speed up inference without giving up correctness or completeness.

We illustrate our method by studying the approximation of logical theories by Horn theories. Following the formal definition of Horn approximation, we present "anytime" algorithms for generating such approximations. We subsequently discuss extensions to other useful classes of approximations.

This paper appears in the *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI–91)*, Anaheim, CA, July 1991.

1 Introduction

The study of the computational properties of knowledge representation systems has revealed a direct trade-off between tractability and expressiveness [Levesque and Brachman, 1985]. In general, in order to obtain a computationally efficient representation system one either restricts the expressive power of the knowledge representation language or one uses an incomplete inference mechanism. In the first approach, the representation language is often too limited for practical applications [Doyle and Patil, 1991]. The second approach involves either resource-bounded reasoning or the introduction of a non-traditional semantics. In resource-bounded reasoning, inference is limited by bounding the number of inference steps performed by the inference procedure. It therefore becomes difficult to characterize exactly what can and cannot be inferred, that is, the approach lacks a "real" semantics (one that does not simply mimic the proof theory). Moreover, no information is provided if a proof cannot be found in the time bound. (But see [Horvitz *et al.*, 1989] for an example of *probabilistic* inference, where confidence in the results increases with the amount of computation.) Accounts of limited inference based on non-traditional semantics [Frisch, 1985; Patel-Schneider, 1986] often provide only a very weak kind of inference. For example, in the four-valued semantics approach of Patel-Schneider, given the statements p and $p \supset q$, one cannot infer q.

This paper presents a third alternative for obtaining efficient representation systems, which neither limits the expressive power of the representation language, nor gives up completeness of the inference procedure. In this new approach, the user enters statements in a general, unrestricted representation language, which the system compiles into a restricted language that allows for efficient inference. Since an exact translation into a tractable form is often impossible, the system searches for the best approximation of the original information. We describe how the approximation can be used to speed up inference. We refer to this approach as *knowledge compilation.*

We illustrate our method by studying the approximation of propositional logic theories by Horn theories. Inference based on these approximations is very fast (linear time). Following the formal definition of Horn approximation, we present algorithms for generating such approximations. The algorithms have the important property that they can be interrupted at any time to produce some useful intermediate result. The paper concludes with a discussion of extensions to first-order theories and various generalizations of Horn

approximations.

2 Horn Approximations

In this section, we introduce the idea of knowledge compilation using a concrete example. We show how a logical theory can be compiled into a tractable form consisting of a set of Horn clauses. It is well-known that in the propositional case reasoning with Horn theories is efficient. Moreover, the experience with logic programming and databases suggests that there are also computational advantages to the use of Horn clauses in the first-order case. We first restrict our attention to the propositional case, and later, briefly consider the first-order case.

We assume a standard propositional language, and use a, b, c, p, q, and r to denote propositional letters and x, y, and z to denote literals (a *literal* is either a propositional letter, called a positive literal, or its negation, called a negative literal). A *clause* is a disjunction of literals, and can be represented by the set of literals it contains. A clause is *Horn* if and only if it contains at most one positive literal; a set of such clauses is called a *Horn theory*. Formulas are given in conjunctive normal form (CNF, a conjunction of disjuncts), so they can be represented by a set of clauses. CNF notation and clausal notation are used interchangeably. For example, we may write $(p \wedge \neg q) \wedge r$ instead of $\{\{p, \neg q\}, \{r\}\}$, and vice versa.

In general, determining whether a given CNF formula (the *query*) follows from a set of formulas in a knowledge base is intractable [Cook, 1971]. However, when the knowledge base contains only Horn clauses the problem can be solved in time linear in the length of the knowledge base combined with the query [Dowling and Gallier, 1984].

So, a useful kind of *knowledge compilation* would be the following: Given a set of arbitrary clauses, compute a logically equivalent set of Horn clauses, and base subsequent inference on that set. Unfortunately, there does not always exist a logically equivalent Horn theory: for example, no Horn theory is equivalent to the theory $p \vee q$. We therefore propose to *approximate* the original set of clauses by a set of Horn clauses. The basic idea is to bound the set of models (satisfying truth assignments) of the original theory from below and from above by Horn theories. In the following definition, $\mathcal{M}(\Sigma)$ denotes the set of satisfying truth assignments of the theory Σ.

Definition: Horn lower-bound and Horn upper-bound
Let Σ be a set of clauses. The sets Σ_{lb} and Σ_{ub} of Horn clauses are respectively

a Horn lower-bound and a Horn upper-bound of Σ iff

$$\mathcal{M}(\Sigma_{\text{lb}}) \subseteq \mathcal{M}(\Sigma) \subseteq \mathcal{M}(\Sigma_{\text{ub}})$$

or, equivalently,

$$\Sigma_{\text{lb}} \models \Sigma \models \Sigma_{\text{ub}}$$

Note that the bounds are defined in terms of models: the lower bounds have fewer models than the original theory, and the upper bound has more models. The reader is cautioned not to associate "lower" with "logically weaker." In fact, because the lower bound has fewer models, it is logically stronger than (*i.e.,* implies) the original theory. Similarly, because the upper bound has more models, it is logically weaker than (*i.e.,* is implied by) the original theory.

Instead of simply using any pair of bounds to characterize the initial theory, we wish to use the best possible ones: a *greatest* Horn lower-bound and a *least* Horn upper-bound.

Definition: Greatest Horn lower-bound (GLB)
Let Σ be a set of clauses. The set Σ_{glb} of Horn clauses is a greatest Horn lower-bound of Σ iff $\mathcal{M}(\Sigma_{\text{glb}}) \subseteq \mathcal{M}(\Sigma)$ and there is no set Σ' of Horn clauses such that $\mathcal{M}(\Sigma_{\text{glb}}) \subset \mathcal{M}(\Sigma') \subseteq \mathcal{M}(\Sigma)$.

Definition: Least Horn upper-bound (LUB)
Let Σ be a set of clauses. The set Σ_{lub} of Horn clauses is a least Horn upper-bound of Σ iff $\mathcal{M}(\Sigma) \subseteq \mathcal{M}(\Sigma_{\text{lub}})$ and there is no set Σ' of Horn clauses such that $\mathcal{M}(\Sigma) \subseteq \mathcal{M}(\Sigma') \subset \mathcal{M}(\Sigma_{\text{lub}})$.

We call these bounds *Horn approximations* of the original theory Σ. The definition of Horn upper-bound implies that the conjunction of two such bounds is another, possibly smaller upper-bound. It follows that a given theory has a unique LUB (up to logical equivalence). On the other hand, a theory can have many different GLBs.

Example: Consider the non-Horn theory $\Sigma = (\neg a \vee c) \wedge (\neg b \vee c) \wedge (a \vee b)$. The Horn theory $a \wedge b \wedge c$ is an example of a Horn lower-bound; both $a \wedge c$ and $b \wedge c$ are GLBs; $(\neg a \vee c) \wedge (\neg b \vee c)$ is an example of a Horn upper-bound; and c is the LUB. The reader can verify these bounds by noting that

$$(a \wedge b \wedge c) \models (a \wedge c) \models \Sigma \models c \models ((\neg a \vee c) \wedge (\neg b \vee c))$$

Moreover, there is no Horn theory Σ' different from $a \wedge c$ such that $(a \wedge c) \models \Sigma' \models \Sigma$. Similar properties hold of the other GLB and of the LUB.

Before we discuss how to compute Horn approximations, let us consider how these approximations can be used to improve the efficiency of a knowledge representation system. Suppose a knowledge base (KB) contains the set of clauses Σ, and we want to determine whether the formula α is implied by the KB. We assume that α is in CNF, because one can determine in linear time if a propositional CNF formula follows from a Horn theory. (Note that the query need not be Horn.) The system can proceed as follows. First, it tries to obtain an answer quickly by using the Horn approximations. If $\Sigma_{\text{lub}} \models \alpha$ then it returns "yes," or if $\Sigma_{\text{glb}} \not\models \alpha$ then it returns "no." So far, the procedure takes only linear-time in the length of the approximations.[1] In case no answer is obtained, the system could simply return "don't know," or it could decide to spend more time and use a general inference procedure to determine the answer directly from the original theory. The general inference procedure could still use the approximations to prune its search space. Thus the system can answer certain queries in linear time, resulting in a improvement in its overall response time. Exactly how many queries can be handled directly by the Horn approximations depends on how well the bounds characterize the original theory. Note that we give up neither soundness nor completeness, because we can always fall back to the original theory.

3 Computing Horn Approximations

We now turn to the problem of generating Horn approximations. Note that there cannot be a polynomial time procedure for generating such approximations (provided P$\neq$NP). This is a direct consequence of the following theorem.

Theorem 1 *Let Σ be a set of clauses. The GLB of Σ is consistent iff Σ is consistent. Similarly, the LUB of Σ is consistent iff Σ is consistent.*

Proof: If $\mathcal{M}(\Sigma) = \emptyset$, then an inconsistent Horn theory such as $p \wedge \neg p$ is both a GLB and LUB of Σ. Let Σ be consistent and M be a satisfying truth assignment of Σ. By definition, M must be a satisfying assignment of any Horn upper-bound. Moreover, the theory with M as the only satisfying truth assignment is a better (larger) Horn lower-bound than an unsatisfiable (inconsistent) Horn theory. So, both the LUB and the GLB of Σ are consistent since each has at least one satisfying truth assignment. ■

[1] We assume that the lengths of the Horn approximations are roughly the same as that of the original theory. We will return to this issue later on.

If the length of the Horn approximations is bounded by some polynomial function of the length of Σ, then the task of finding them is NP-hard, because checking the consistency of a general set of clauses is NP-complete, but checking the consistency of Horn clauses takes only linear-time. On the other hand, if certain approximations are of exponential length, then it certainly takes exponential time to generate them. So, in either case the problem is intractable. Of course, if the approximations were polynomial time computable, then they could not be very good approximations (for example, inconsistency could go undetected), and at query time they could save at most a polynomial amount of time on an exponentially hard problem.

Computing the Horn approximations should be viewed as a compilation process. The computational cost can be amortized over the total set of subsequent queries to the KB. In some cases, however, the approximations may be needed for query answering before the compilation process finishes. So, instead of waiting for the best Horn bounds, it would be desirable to employ procedures that could output lower- and upper-bounds as intermediate results, generating better and better bounds over time. That is, the approximation algorithms should be "anytime" procedures [Boddy and Dean, 1988]. The algorithms presented in this paper have this property.

We discuss a method for generating the GLB first. The following notion is central to our approach:

Definition: Horn-strengthening
A Horn clause $C_{\rm H}$ is a Horn-strengthening of a clause C iff $C_{\rm H} \subseteq C$ and there is no Horn clause $C'_{\rm H}$ such that $C_{\rm H} \subset C'_{\rm H} \subseteq C$.

Example: Consider the clause $C = \{p, q, \neg r\}$. The clauses $\{p, \neg r\}$ and $\{q, \neg r\}$ are Horn-strengthenings of C.

The following two lemmas state some useful properties of Horn-strengthenings. The first lemma shows that a Horn theory entails a clause only if it entails some Horn-strengthening of the clause.

Lemma 1 *Let $\Sigma_{\rm H}$ be a Horn theory and C a clause that is not a tautology. If $\Sigma_{\rm H} \models C$ then there is a clause $C_{\rm H}$ that is a Horn-strengthening of C such that $\Sigma_{\rm H} \models C_{\rm H}$.*

Proof: By the subsumption theorem [Lee, 1967], there is a clause C' that follows from $\Sigma_{\rm H}$ by resolution such that C' subsumes C. Because the resolvent of Horn clauses is Horn, C' is Horn, and thus is a Horn-strengthening of C. ■

GLB Algorithm
Input: a set of clauses $\Sigma = \{C_1, C_2, \ldots, C_n\}$.
Output: a greatest Horn lower-bound of Σ.
begin
    $L :=$ the lexicographically first Horn–
        strengthening of Σ
    loop
        $L' :=$ lexicographically next Horn–
            strengthening of Σ
        if none exists **then exit**
        if $L \models L'$ **then** $L := L'$
    end loop
    remove subsumed clauses from L
    return L
end

Figure 1: Algorithm for generating a greatest Horn lower-bound.

The next lemma shows that every GLB of a theory — given in clausal form — is equivalent to some subset of the Horn-strengthenings of the clauses of the theory.

Lemma 2 *If Σ_{glb} is a GLB of a theory $\Sigma = \{C_1, \ldots, C_n\}$, then there is a set of clauses $C'_1, \ldots, C'_n$ such that $\Sigma_{\text{glb}} = \{C'_1, \ldots, C'_n\}$, where C'_i is a Horn-strengthening of C_i.*

Proof: Directly from lemma 1 and the definition of GLB. ∎

So, each GLB of Σ is given by some Horn-strengthening of the clauses of Σ. Moreover, it is not difficult to see that the set containing a Horn-strengthening of each clause is a Horn lower-bound of Σ — though not necessarily the greatest lower-bound. This leads us to the **GLB algorithm** given in figure 1. The algorithm systematically searches through the various possible Horn-strengthenings of the clauses of the original theory, looking for a most general one. Where $\Sigma = \{C_1, C_2, \ldots, C_n\}$ and C_i^j is the j-th Horn strengthening of clause C_i, the Horn strengthenings of Σ are generated in the lexicographic order $\{C_1^1, C_2^1, \ldots, C_n^1\}$, $\{C_1^2, C_2^1, \ldots, C_n^1\}$, $\{C_1^1, C_2^2, \ldots, C_n^1\}$, $\{C_1^2, C_2^2, \ldots, C_n^1\}$, etc.

Theorem 2 *Given a set of clauses Σ, the GLB algorithm (Fig. 1) computes a greatest Horn lower-bound of Σ of length less than or equal to that of Σ.*

Example: Consider the theory $\Sigma = (\neg a \vee b \vee c) \wedge (a \vee b)$. The algorithm first tries the Horn-strengthening $L = ((\neg a \vee b) \wedge a) \equiv (a \wedge b)$, and then $L' = ((\neg a \vee b) \wedge b) \equiv b$. Since $L \models L'$, L is set to L', and the algorithm proceeds. Since the other two Horn strengthenings do not further improve the bound, the algorithm returns b as an answer (the redundant clause $(\neg a \vee b)$ is removed by the last step of the algorithm).

Proof of correctness of the GLB algorithm: Assume that the set of clauses L returned by the algorithm is not a GLB of Σ. Clearly, L is a Horn lower-bound. So, it follows that there is some greatest Horn lower-bound L' of Σ such that $L \models L' \models \Sigma$ and $L' \not\models L$. By lemma 2, L' is equivalent to some Horn-strengthening $L^\star$ of the clauses of Σ. So, the algorithm would have returned $L^\star$. Contradiction. ■

The **GLB algorithm** is indeed an anytime algorithm: L represents some lower-bound whenever the algorithm is interrupted. Note also that the total running time of the algorithm is exponential only in the length of the non-Horn part of the original theory, because the only strengthening of a Horn clause is the clause itself.

A theory may have exponentially many greatest Horn lower-bounds, so in practice one would not want to generate them all. However, there is a simple and significant relationship between a theory and the set of all its GLBs, which follows from the fact that each model of a theory is a lower-bound of the theory:

Theorem 3 *Let Σ be a set of clauses. Then Σ is logically equivalent to the disjunction of all the greatest Horn lower-bounds of Σ.*

We now turn our attention to the generation of the LUB. We will use the notion of a *prime implicate* of a theory, which is a strongest clause implied by the theory. The following theorem reveals our basic strategy.

Theorem 4 *Let Σ be a set of clauses. The LUB of Σ is logically equivalent to the set of all Horn prime implicates of Σ.*

Proof: The set of Horn prime implicates is implied by Σ, and thus is a Horn upper-bound. Furthermore, it must the LUB, because at least one of its clauses subsumes (and therefore implies) any clause in any Horn upper-bound. ■

LUB Algorithm
Input: a set of clauses $\Sigma = \Sigma_{\rm H} \cup \Sigma_{\rm N}$, where
$\Sigma_{\rm H}$ is a set of Horn clauses, and $\Sigma_{\rm N}$ is a
set of non-Horn clauses.
Output: a least Horn upper-bound of Σ.
begin
loop
try to choose clause $C_0 \in \Sigma_{\rm H} \cup \Sigma_{\rm N}$ and
$C_1 \in \Sigma_{\rm N}$, such that $C_2 = {\rm Resolve}(C_0, C_1)$
is not subsumed by any clause in $\Sigma_{\rm H} \cup \Sigma_{\rm N}$
if no such choice is possible **then exit loop**
if C_2 is Horn **then**
delete from $\Sigma_{\rm H}$ and $\Sigma_{\rm N}$ any clauses
subsumed by C_2
$\Sigma_{\rm H} := \Sigma_{\rm H} \cup \{C_2\}$
else
delete from $\Sigma_{\rm N}$ any clauses subsumed by C_2
$\Sigma_{\rm N} := \Sigma_{\rm N} \cup \{C_2\}$
end if
end loop
return $\Sigma_{\rm H}$
end

Figure 2: Algorithm for generating a least Horn upper-bound.

So, in principle, we could use resolution to generate the prime implicates and simply collect the Horn ones in order to generate the least Horn upper-bound. However, such a method could prove very expensive since even if the original theory contains only Horn clauses, there can be exponentially many Horn resolvents (for an example, see Selman [1990].) Clearly, such resolvents add nothing to the best approximation of the original Horn theory, since the least Horn upper-bound is already given by the theory itself. Fortunately, we can improve upon the procedure of generating all prime implicates by only resolving two clauses if at least one of them is non-Horn.

Theorem 5 *Given a set of clauses Σ, the LUB algorithm (Fig. 2) computes the least Horn upper-bound of Σ.*

Example: Consider the theory $(\neg a \vee b) \wedge (\neg b \vee c) \wedge (a \vee b)$. The **LUB algorithm** resolves the first and the third clause, obtaining the clause b. Σ_{H} becomes $(\neg b \vee c) \wedge b$, upon which the loop is exited and Σ_{H} is returned.

The correctness proof of the **LUB algorithm** [Selman and Kautz, 1991] is quite involved and we therefore do not include it here. As with the GLB, the algorithm is anytime: Σ_{H} improves over time.

Finally, we briefly consider the size of the generated Horn approximations. From the **GLB algorithm** it follows immediately that the size of the generated bound is less than or equal to that of the original theory. So, the system can safely use this bound; even if the approximation does not provide an answer for a particular query, at most a linear amount of time in terms of the length of the original theory is being "wasted." Whether there always exists a similarly short least Horn upper-bound is currently an open question. We conjecture that there are cases where the **LUB algorithm** generates a bound that is exponentially longer then the original theory, although we expect the blow-up to be exponential only in the size of the non-Horn part of the theory. In such cases, the system has to somehow limit the length of the approximation, for example, by using a weaker bound or by minimizing the length of the bound (*e.g.*, replace $(a \vee \neg b) \wedge b$ by $a \wedge b$).

4 Relation to Other Approaches

In this section, we discuss some related work. The Horn upper-bound can be viewed as a generalization or abstraction of the original theory, because the upper-bound is a more general, weaker theory. To illustrate this, we now show how the notion of abstraction as introduced by Borgida and Etherington [1989] directly corresponds to the least Horn upper-bound of the theories that they consider.

Borgida and Etherington propose using background knowledge that captures the hierarchical relationship among predicates in order to replace disjunctions by more general concepts. Suppose the background knowledge is `doctor(Jill)` $\supset$ `professional(Jill)` and `lawyer(Jill)` $\supset$ `professional(Jill)`, and the KB is `doctor(Jill)` $\vee$ `laywer(Jill)`. They then generate a new KB that contains only `professional(Jill)`. But this is simply the least Horn upper-bound of the original KB together with the background knowledge. Note that the idea of an LUB is much more general than Borgida and Etherington's approach, since it can be applied to arbitrary propositional Horn

theories — not just concept hierarchies with positive disjunctions. However, their specialized procedures may run faster than our more general ones.

A Horn lower-bound corresponds to a more specific theory than the original one. Its use generalizes the use of a counterexample to prune the search space of inference procedures. The best-known example of the use of counterexamples in artificial intelligence (AI) can be found in the early work by Gelernter [1959] on proving theorems in geometry. Gelernter used a single model M (given by a diagram) of the original theory Σ to answer certain queries negatively, based on the fact that if $M \not\models \alpha$ then $\Sigma \not\models \alpha$, for a query α. The Horn lower-bound is used in a similar manner, but it will generally involve a *set* of models, and is thus a better characterization of the original theory. In particular, one may avoid some of the "accidental truths" that often hold in a single model or diagram.

The work in AI on problem solving with abstraction [Amarel, 1968; Sacerdoti, 1974; Plaisted, 1981] is less directly related to the knowledge-compilation approach. In the work on abstraction one maps a theory to a smaller theory, generates proofs in the smaller theory, and then uses the proofs to guide generation of proofs in the original theory. While the idea of transforming a theory in order to make inference faster is similar, the abstraction approach does not in general preserve consistency in the sense of Theorem 1 (but see Tenenberg [1988] for an exception), and does not map theories into a particular syntactic form that is guaranteed to be efficient. Most importantly, the abstraction mappings are supplied by the user and are domain-specific; in contrast, the Horn approximations in our approach are generated automatically.

5 Other Kinds of Approximations

While we have concentrated on propositional Horn approximations, the general knowledge compilation approach can be applied to produce any efficient logical form. In this section, we briefly discuss some of the other kinds of approximation we are investigating.

One natural extension of this work is to generate first-order Horn approximations of first-order theories. As noted earlier, while first-order Horn theories are not necessarily tractable, in practice they tend to support fast inference. Because satisfiability is undecidable for first-order languages, it is clear that there cannot be algorithms to generate approximations that always terminate. However, we also noted that the algorithms presented in this paper are anytime algorithms, so they can be used (with minor modifications [Selman and Kautz,

1991]) to generate an infinite sequence of better and better bounds. One open technical question is whether first-order clausal theories always have a *greatest* Horn lower-bound. **(Note: they do; see [Selman and Kautz, 1991].)**

We are also investigating the general classes of approximations that can be generated using techniques based on those described in this paper [Kautz and Selman, 1991]. At least in the propositional case, the GLB and LUB algorithms can be modified to generate approximations that are in any class of clauses with the following properties: the class is closed under resolution, closed under subsumption, and any general clause is subsumed by some clause in the class. A simple but useful example of such a class is all clauses *not* containing a given set of predicates; the techniques can thus be used to "compile away" a set of "irrelevant" predicates [Subramanian and Genesereth, 1987]. Another useful case is the class of binary clauses, since satisfiability can be determined in polynomial time.

6 Conclusions

We introduced the notion of knowledge compilation. The basic idea is to compile knowledge from an intractable into a tractable form. Since an exact translation is often not possible, we introduced approximate translations, consisting of two bounds that delimit the original theory.

Knowledge compilation provides an alternative to approaches that force the user to state all knowledge in some restricted but tractable language. A representation system incorporating a knowledge compilation procedure will allow the user to enter information in a general, unrestricted language, while the system compiles such information into a tractable form.

To illustrate our approach, we showed how knowledge represented in a propositional theory can be approximated using two Horn theories, called Horn approximations: a greatest Horn lower-bound and a least Horn upper-bound. Answering a query based on the original knowledge base is intractable, but by using the Horn approximations certain queries can be answered in time linear in the length of the approximations. We gave algorithms for generating such Horn approximations. The algorithms operate incrementally, generating better and better approximations over time. The incremental nature of the approximation algorithms is a key feature of our approach, since in practical applications it would be unacceptable to have to wait until the system has computed the best bounds before answering any queries.

In summary, the main features of our knowledge compilation approach are:

- A guaranteed fast response for queries that can be answered directly using the approximations.
- An incremental, off-line compilation process that provides continuous improvement of the overall response time of the system.
- No loss of soundness or completeness.

We also considered some generalizations of the Horn approximation notion and discussed its relationship with research on abstraction in problem-solving.

Acknowledgements

We thank Daniel Bobrow for getting us to think more about the issue of how to make practical use of restricted, tractable representation languages and Ray Reiter for pointing us to the work on prime-implicates. We also thank Hector Levesque and Yoav Shoham for useful comments and discussions.

References

[Amarel, 1968] Saul Amarel. On representations of problems of reasoning about actions. In Michie, editor, *Machine Intelligence 3*, pages 131–171. Edinburgh University Press, 1968.

[Boddy and Dean, 1988] Mark Boddy and Thomas Dean. Solving time dependent planning problems. Technical report, Department of Computer Science, Brown University, 1988.

[Borgida and Etherington, 1989] Alex Borgida and David W. Etherington. Hierarchical knowledge bases and efficient disjunctive reasoning. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 33–43, Toronto, Ontario, 1989. Morgan Kaufmann Publishers, Inc.

[Cook, 1971] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing*, pages 151–158, 1971.

[Dowling and Gallier, 1984] William F. Dowling and Jean H. Gallier. Linear time algorithms for testing the satisfiability of propositional horn formula. *Journal of Logic Programming*, 3:267–284, 1984.

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

[Frisch, 1985] Alan M. Frisch. Using model theory to specify AI programs. In *Proceedings of IJCAI-85*, pages 148–154, 1985.

[Gelernter, 1959] H. Gelernter. Realization of a geometry theorem-proving machine. In *Proceedings of the International Conference on Information Processing*, pages 273–282, Paris, 1959. UNESCO House. (Reprinted in *Computers and Thought*, E. Feigenbaum and J. Feldman (Eds.), McGraw-Hill, NY, pages 134-152, 1963.).

[Horvitz *et al.*, 1989] Eric J. Horvitz, Gregory F. Cooper, and David E. Heckerman. Reflection and action under scarce resources: Theoretical principles and empirical study. In *Proceedings of IJCAI-89*, page 1121, Detroit, MI, May 1989. Morgan Kaufmann.

[Kautz and Selman, 1991] Henry Kautz and Bart Selman. A general framework for knowledge compilation, 1991. Submitted for publication.

[Lee, 1967] R. C. T. Lee. *A Completeness Theorem and a Computer Program for Finding Theorems Derivable From Given Axioms*. PhD thesis, University of California at Berkeley, Berkeley, CA, 1967.

[Levesque and Brachman, 1985] H.J. Levesque and R.J. Brachman. A fundamental tradeoff in knowledge representation and reasoning (revised version). In R.J. Brachman and H.J. Levesque, editors, *Readings in Knowledge Representation*, pages 41–70. Morgan Kaufmann, Los Altos, CA, 1985.

[Patel-Schneider, 1986] Peter F. Patel-Schneider. A four-valued semantics for frame-based description languages. In *Proceedings of AAAI-86*, pages 344–348, Philadelphia, PA, 1986.

[Plaisted, 1981] D. Plaisted. Theorem proving with abstraction. *Artificial Intelligence*, 16:47, 1981.

[Sacerdoti, 1974] Earl D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5(2):115–135, 1974.

[Selman and Kautz, 1991] Bart Selman and Henry Kautz. Methods of knowledge compilation. In Preparation, 1991.

[Selman, 1990] Bart Selman. Tractable default reasoning. Ph.D. Thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, 1990.

[Subramanian and Genesereth, 1987] Devika Subramanian and Michael R. Genesereth. The relevance of irrelevance. In *Proceedings of IJCAI-87*, volume 1, page 416, 1987.

[Tenenberg, 1988] Josh D. Tenenberg. Abstraction in planning. Technical Report 250, Computer Science Department, University of Rochester, Rochester, NY, May 1988.