Journal of the ACM, 43(2):193-224, 1996.

Knowledge Compilation and Theory Approximation

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

Abstract

Computational efficiency is a central concern in the design of knowledge representation systems. In order to obtain efficient systems, it has been suggested that one should limit the form of the statements in the knowledge base or use an incomplete inference mechanism. The former approach is often too restrictive for practical applications, whereas the latter leads to uncertainty about exactly what can and cannot be inferred from the knowledge base. We present a third alternative, in which knowledge given in a general representation language is translated (compiled) into a tractable form — allowing for efficient subsequent query answering.

We show how propositional logical theories can be compiled into Horn theories that approximate the original information. The approximations bound the original theory from below and above in terms of logical strength. The procedures are extended to other tractable languages (for example, binary clauses) and to the first-order case. Finally, we demonstrate the generality of our approach by compiling concept descriptions in a general frame-based language into a tractable form.

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 within 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 [Levesque, 1984, Frisch, 1985, Patel-Schneider, 1986] often provide only a very weak kind of inference. For example, in the four-valued semantics approach of Levesque, 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. Applications of propositional inference arise in many areas of artificial intelligence and computer science in general. Examples include qualitative reasoning about physical systems [Reiter and de Kleer, 1987], circuit diagnosis [Larrabee, 1992], expert systems [Dean, 1985], and learning theory [Valiant, 1983]. In addition, many reasoning tasks can be directly mapped into propositional inference problems, such as finite-domain constraint satisfaction problems (CSP) [Dechter, 1992] and certain classes of planning problems [Kautz and Selman, 1992b].

Finally, note that first-order domain theories in artificial intelligence often involve only finite domains. Such theories are therefore essentially propositional.

For example, Reiter and Mackworth [1989] discuss how their first-order domain theory for visual interpretation can be restated in propositional form. By considering the propositional formulation, they show how methods from relational databases can be used to answer queries much more efficiently than by using a general first-order theorem prover on the original domain theory. Another example is our propositional formulation of planning, cited above.

General propositional reasoning is the prototypical NP-complete task. However, inference based on Horn approximations is very fast (linear time). Following the formal definition of Horn approximation, we present algorithms for generating such approximations. The algorithms have the property that they can be interrupted at any time to produce some useful intermediate result. We also present an empirical evaluation of our approach, and demonstrate substantial computational savings on computationally challenging propositional theories. The paper concludes with a discussion of extensions to first-order theories and various generalizations of Horn approximations, as well as to description logics. A preliminary discussion of these ideas appeared in [Selman and Kautz, 1991] and [Kautz and Selman, 1991].

2 Propositional Horn Approximations

In this section, we introduce the idea of knowledge compilation using a concrete example. We show how a propositional theory can be compiled into a tractable form consisting of a set of Horn clauses.

2.1 Definitions

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 $(p \vee \neg q) \wedge r$ instead of $\{\{p, \neg q\}, \{r\}\}$, and vice versa.

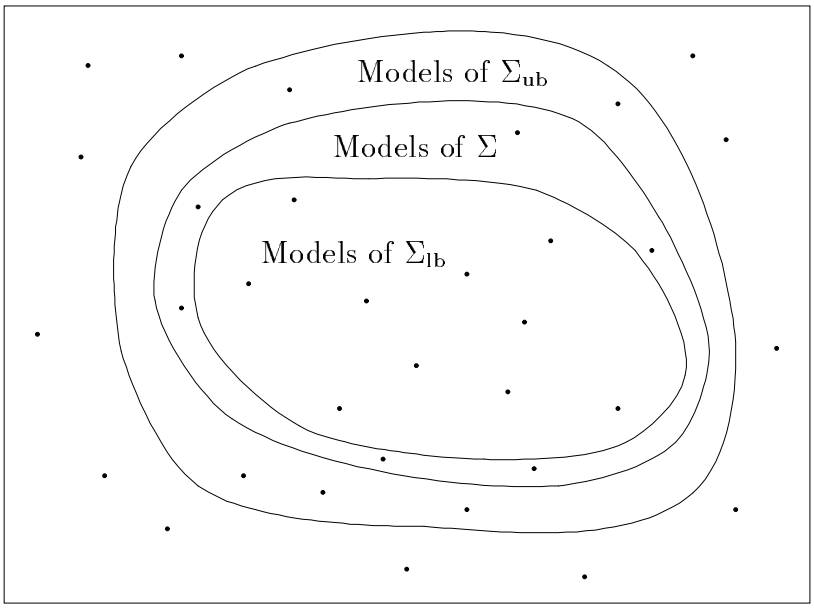

Figure 1: Two sets of models that bound the original set of models of the original theory Σ from below and above. Models are represented by dots. Each of the bounds can be represented by a set of Horn clauses.

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. Fig. 1 illustrates the basic idea.

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 $\Sigma_{\rm lb}$ and $\Sigma_{\rm ub}$ of Horn clauses are respectively a Horn lower-bound and a Horn upper-bound of Σ iff

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

or, equivalently,

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

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 $\Sigma_{\rm glb}$ of Horn clauses is a greatest Horn lower-bound of Σ iff $\mathcal{M}(\Sigma_{\rm glb}) \subseteq \mathcal{M}(\Sigma)$ and there is no set Σ' of Horn clauses such that $\mathcal{M}(\Sigma_{\rm glb}) \subset \mathcal{M}(\Sigma') \subseteq \mathcal{M}(\Sigma)$.

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

We call these bounds *Horn approximations* of the original theory Σ.[1] The definition of a 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

[1] We will sometimes use the term Horn approximation to refer to *any* Horn upper or lower-bound, not necessarily the LUB or the GLB.

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' logically distinct from $a \wedge c$ such that $(a \wedge c) \models \Sigma' \models \Sigma$. Similar properties hold of the other GLB and of the LUB.

2.2 Using Approximations For Fast Inference

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 shown in Fig. 2. First, it tries to obtain an answer quickly by using the Horn approximations. If $\Sigma_{\rm lub} \models \alpha$ then it returns "yes," or if $\Sigma_{\rm glb} \not\models \alpha$ then it returns "no." So far, the procedure takes only time linear in the length of the approximations.[2] 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. (We will return to this issue in Section 2.4 below.) Note that we give up neither soundness nor completeness, because we can always fall back to the original theory.

When the system opts for returning "don't know" instead of performing general inference with the original theory, we have an example of trading accuracy

[2] 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.

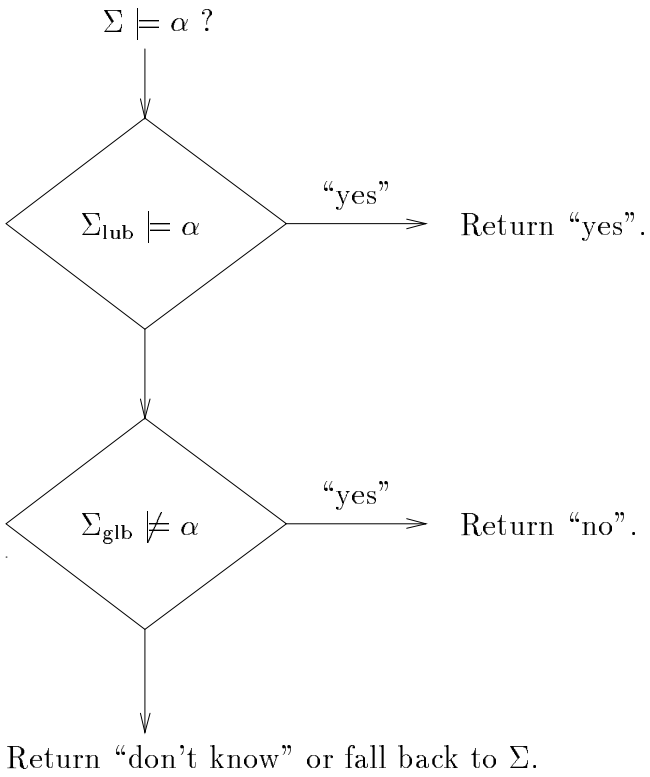

Figure 2: Fast querying using theory approximation. The original theory is Σ; Σ_{lub} and Σ_{glb} are its approximations; and α is the query.

for speed. This approach may be quite useful in real-time systems that have some fall back mechanism for dealing with the "don't know" answers. For example, if it too expensive to answer a query by inference, a robot may instead try to determine the answer by sensing.

This paper concentrates on compiling a given knowledge base before any queries are made. Greiner [1992] and Greiner and Schuurmans [1992] extend our framework by providing compilation algorithms that interleave compilation and query answering. Another interesting direction for future research would be to allow for updates to the knowledge base between queries.

2.3 Computing Horn Approximations

We now turn to the problem of generating Horn approximations. There does not exist a polynomial time procedure for generating such approximations (provided P$\neq$NP). This is 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. ■

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, whereas 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. Thus, 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 $C_{\rm H} \subset C'_{\rm H} \subseteq C$. A Horn-strengthening of a set of clauses $\{C_1, \ldots, C_n\}$ is any set of clauses $\{C'_1, \ldots, C'_n\}$ where C'_i is a Horn-strengthening of C_i.

Thus, a Horn-strengthening of a non-Horn clause is formed by removing all but one positive literal.
Example: Consider the clause $C = \{p, q, \neg r\}$. The clauses $\{p, \neg r\}$ and $\{q, \neg r\}$ are the 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 $\Sigma_{\rm H} \models C_{\rm H}$. More generally, if a Horn theory entails a set of clauses, then it entails a Horn-strengthening of the set.*

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. Therefore there is some $C_{\rm H}$ that is a Horn-strengthening of C such that $C' \subseteq C_{\rm H} \subseteq C$, and $\Sigma_{\rm H} \models C_{\rm H}$. The generalization to sets of clauses follows immediately. ■

The next lemma shows that every GLB of a theory is equivalent to some Horn-strengthening of the theory.

Generate_GLB
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 loop**
        if $L \models L'$ **then** $L := L'$
    end loop
    remove subsumed clauses from L
    return L
end

Figure 3: Algorithm for generating a greatest Horn lower-bound.

Lemma 2 *Let Σ_{glb} be 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 $\Sigma_{\text{glb}} \equiv \{C'_1, \ldots, C'_n\}$, where C'_i is a Horn-strengthening of C_i.*

Proof: By the definition of a GLB, we have $\Sigma_{\text{glb}} \models \Sigma$. Therefore $\Sigma_{\text{glb}} \models C_i$ for $1 \leq i \leq n$. Because Σ_{glb} is Horn, by Lemma 1, there exists $\{C'_1, \ldots, C'_n\}$, where each C'_i is a Horn-strengthening of C_i, such that $\Sigma_{\text{glb}} \models C'_i$ for $1 \leq i \leq n$. Thus $\Sigma_{\text{glb}} \models \{C'_1, \ldots, C'_n\} \models \Sigma$. Moreover, because Σ_{glb} is a *greatest* lower-bound, it must be the case that $\Sigma_{\text{glb}} \equiv \{C'_1, \ldots, C'_n\}$. ■

Moreover, it is not difficult to see that every Horn-strengthening of Σ is a Horn lower-bound of Σ — though not necessarily the greatest lower-bound. This leads us to the **Generate_GLB** algorithm given in Fig. 3. 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,* **Generate_GLB** *(Fig. 3) 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 Theorem 2 Ignoring the last step that removes subsumed clauses from L, it is obvious that **Generate_GLB** returns a logically-weakest Horn-strengthening of Σ. So to prove the algorithm correct, we need to show that any logically-weakest Horn-strengthening of Σ is a GLB of Σ. First note that $L \models \Sigma$ and L is Horn so L is a LB. Suppose it were not a greatest LB. There is a set of Horn clauses H such that

$$\begin{aligned} L &\models H \\ H &\not\models L \\ H &\models \Sigma \end{aligned}$$

By Lemma 1 there must be some Horn strengthening Σ' of Σ such that

$$H \models \Sigma'$$

But this means that

$$L \models \Sigma'$$

Now there are two cases: (1) L and Σ' are the logically equivalent. Then this means that L and H are logically equivalent, which is a contradiction. (2) L and Σ' are logically distinct. But then L is not a logically-weakest Horn strengthening of Σ, which is also a contradiction. This proves the theorem. ■

Generate_GLB 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.

The querying algorithm shown in Fig. 2 uses only a single GLB of the theory. One could extend the algorithm to use several lower bounds (although not necessarily all of them, because certain theories can have exponentially many distinct

GLBs). There is a simple and interesting relationship between a theory and the set of all of its GLBs. This follows from the fact that each model of a theory corresponds to a a lower-bound of the theory, namely the conjunction of all the literals assigned true by the model. We thus obtain the following theorem:

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.*[3]

Proof: The set of Horn prime implicates is implied by Σ, and thus is a Horn upper-bound. Furthermore, it must be the LUB, because at least one of its clauses subsumes (and therefore implies) any clause in any 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 is non-Horn. The intuition behind this improvement is that any resolution proof tree that ends in a Horn clause can be rewritten so that all the resolutions between pairs of Horn clauses appear at the bottom of the tree. The compilation phase essentially generates the top part of the tree, and the all-Horn part is generated at query-answering time. The full inductive proof that this method is complete was provided by del Val [1995] . As with **Generate_GLB**, the algorithm is anytime: Σ_{H} improves over time.

Theorem 5 *Given a set of clauses Σ,* **Generate_LUB** *algorithm (Fig. 4) computes the least Horn upper-bound of Σ.*

[3] It follows directly that the LUB is complete with respect to Horn queries. That is, for any Horn clause α, we have $\Sigma \models \alpha$ if and only if $\Sigma_{\mathrm{lub}} \models \alpha$.

Generate_LUB
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 $C_0 \in \Sigma_{\rm H}$ and $C_1 \in \Sigma_{\rm H} \cup \Sigma_{\rm N}$,
such that $C_2 =$ 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**
delete from $\Sigma_{\rm H}$ and $\Sigma_{\rm N}$ any clauses
subsumed by C_2
if C_2 is Horn **then**
$\Sigma_{\rm H} := \Sigma_{\rm H} \cup \{C_2\}$
else
$\Sigma_{\rm N} := \Sigma_{\rm N} \cup \{C_2\}$
end if
end loop
return $\Sigma_{\rm H}$
end

Figure 4: Algorithm for generating the least Horn upper-bound.

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.

2.4 Computational Savings

In Section 2.2 we described how the bounds can be used to answer queries, and claimed that doing so could lead to computational savings. One might wonder whether it could be the case that queries that can be answered by the bounds are also easy to answer using the original theory. A central aspect of our approach is that this is *not* so. An obvious counterexample is any inconsistent theory. Compilation yields an inconsistent upper bound. Any query against this bound would quickly return "yes" (see Fig. 2). However, evaluating a query against the original theory would in general involve proving that the theory was inconsistent, which is NP-complete.

Of course, most interesting knowledge bases will be consistent. Let us therefore consider a consistent theory that is equivalent to a Horn theory, but is not in Horn form. Clearly, all queries can be answered efficiently against the bounds. However, it is *not* the case that a theorem prover could also answer queries efficiently against the original theory. This can be shown using a result by Valiant and Vazirani [1986] . They show that even if a propositional theory has a single model (and is thus trivially equivalent to a Horn theory), finding the model is still intractable (unless NP $\neq$ RP, which is unlikely). Therefore, there cannot exist a theorem prover that efficiently handles this special case, because such a prover could be used to find the unique model of the non-Horn theory (by repeatedly testing whether each literal followed from the theory).

The fact that the bounds obtained by our method can be used to answer queries efficiently that would be intractable to test against the original theory is a key feature that distinguishes our work from most related work on reasoning with approximations (related work is discussed in Section 4 below). The most natural application of our framework is therefore in domains where fast response to queries is of the highest importance. Direct practical evidence for this can be found in the work by de Kleer [1990] on qualitative physics. De Kleer uses an Assumption-Based Truth Maintenance System (ATMS), a kind of propositional theorem prover, to reason about logical theories of qualitative physics. The main form of inference of the ATMS is so-called Boolean Constraint Propagation (BCP). BCP is sound

and complete for Horn clauses, but incomplete for more general theories. In order to make BCP more useful, the system preprocesses the theory using a limited form of resolution. For example, given the clauses $x \vee \neg y \vee z$ and $\neg x \vee \neg y \vee z$, one can add their resolvent $\neg y \vee z$. This additional clause will allow BCP to derive additional information. Since BCP in effect ignores non-Horn clauses, the process can be viewed as computing a form of a Horn upper-bound approximation to the original theory. De Kleer's approach is thus a concrete example of the practical usefulness of a limited form of knowledge compilation.[4]

In order to gain some further insight into the practical usefulness of knowledge compilation, we now consider the compilation of hard, randomly-generated propositional theories. Mitchell, Selman and Levesque [1992] show that most randomly-generated theories are easy to reason with. Such theories tend to be either very over-constrained or very under-constrained; in either case, experiments show that answering queries is easy using the standard Davis-Putnam procedure [Davis and Putnam, 1960].[5] However, Mitchell *et al.* also described how to randomly generate computationally challenging theories. The key is to generate formulas with a particular ratio of clauses to variables. For random 3CNF formulas, the ratio is about 4.3. We consider hard random 3CNF theories containing between 75 and 200 variables. In order to simplify the following analysis, we computed bounds that consisted of conjunctions of unit clauses. Note that unit clauses are a restricted case of Horn clauses. Therefore, these bounds are not as tight as the full Horn bounds. We will show that even these bounds are useful for answering a high percentage of queries. Because the full Horn bounds are tighter, they would answer an even higher percentage of queries. However, by considering the unit clause bounds we are able to provide a simple exact analysis.

We began by generating a set of 40 random 3CNF theories, with 10 each based on 75, 100, 150, and 200 variables. Then we computed the unit LUB and a unit

[4] De Kleer also discusses the possibility of replacing the entire theory by its set of prime implicates. Though this may be useful for abductive reasoning [Reiter and de Kleer, 1987, Selman and Levesque, 1990], it is not necessary for deductive reasoning because Horn clauses themselves already allow for efficient inference. Also note that generating all the prime implicates may require exponential space.

[5] If the theory is over-constrained, it is generally unsatisfiable, so that all queries trivially follow. If it is under-constrained and the CNF query contains short disjunctions, then the query almost certainly does not follow. Finally, if the theory is under-constrained and the CNF query contains only long disjunctions, then the query almost certainly does follow, which can be easily shown by adding the negation of the query to the theory and using the Davis-Putnam procedure with unit propagation to show inconsistency.

vars	clauses	size unit LUB	size unit GLB	percent queries answered		
				unit	binary	ternary
75	322	53	71	100%	85%	88%
100	430	57	93	100%	76%	79%
150	645	62	139	100%	66%	66%
200	860	132	188	100%	83%	85%

Table 1: Size of bounds and percentage of random queries answered by unit bounds for hard random 3CNF theories. The size of the unit LUB and unit GLB were determined empirically. The table gives the median values over 10 random theories of each size. The percentage of random queries answered by the bounds was then computed probabilistically, as described in the text.

GLB of each. Table 1 gives the average size, in literals, of the LUB and GLB for each size theory.

We computed the unit upper bound using a version of the **Generate_LUB** algorithm that was restricted to output only unit clauses. **Generate_GLB** was similar adapted for unit bounds; in particular, the first strengthening of the theory was taken to be a satisfying model of the theory found using a fast randomized satisfiability procedure [Selman *et al.*, 1992].

Computation time for the unit LUBs ranged from 5 minutes for the 75 variable theories, to one hour for the 200 variable theories, on a 100Mhz SGI Challenge. Computation of the unit GLBs ranged from 1 minute to 5 minutes each.

The percentage of queries that could be answered by these bounds is computed using some basic probability theory. We assume that we are dealing with single-clause queries drawn from a uniform fixed-clause length distribution. The simpliest case is the unit clause queries. All unit clause queries can be answered using only the unit LUB, because this bound is complete for such queries. Thus this column is 100% for every size theory.

Next, let us consider the more interesting case of binary queries. Let $x \vee y$ be a random binary clause, where x and y are distinct and not complements. We wish to compute the probability that the bounds answer the query, given that the unit LUB is of size l and the unit GLB is of size m, and there are N variables in

the theory. That is, we wish to compute

$$Pr((\Sigma_{\text{lub}} \vdash x \vee y) \text{ or } (\Sigma_{\text{glb}} \nvdash x \vee y))$$

which equals

$$Pr(\Sigma_{\text{lub}} \vdash x \vee y) + Pr(\Sigma_{\text{glb}} \nvdash x \vee y)$$

because the two possibilities are disjoint. A disjunction is entailed by a set of literals if and only if one of the disjuncts is so entailed. Thus,

$$Pr(\Sigma_{\text{lub}} \vdash x \vee y) = Pr((\Sigma_{\text{lub}} \vdash x) \text{ or } (\Sigma_{\text{lub}} \vdash y))$$

Note that we were only able to make this step because we are considering the unit bounds, instead of the full Horn bounds. This quantity is equal to

$$Pr(\Sigma_{\text{lub}} \vdash x) + (\Sigma_{\text{lub}} \vdash y) - (\Sigma_{\text{lub}} \vdash x \wedge y)$$

The first and second terms are equal to the odds of picking a random literal that is in the LUB, namely $l/(2N)$. The third term is equal to the number of ways of choosing two distinct literals from the LUB, divided by the number of ways of choosing two distinct, non-complementary literals, namely $l(l-1)/((2N)2(N-1))$. Thus,

$$Pr(\Sigma_{\text{lub}} \vdash x \vee y) = \frac{l}{2N} + \frac{l}{2N} - \frac{l(l-1)}{4N(N-1)}$$

Next, consider the probability that a binary query is answered by the GLB. We have

$$Pr(\Sigma_{\text{glb}} \nvdash x \vee y) = 1 - Pr(\Sigma_{\text{glb}} \vdash x \vee y)$$

The final term above can be calculated using the same argument as for the LUB to yield

$$Pr(\Sigma_{\text{glb}} \nvdash x \vee y) = 1 - \left(\frac{m}{2N} + \frac{m}{2N} - \frac{m(m-1)}{4N(N-1)} \right)$$

Combining the probability that the LUB answers the query with the probability that the GLB answers the query results in the expression

$$1 - \frac{4N(m-l) - 3(m-l) + l^2 - m^2}{4N(N-1)}$$

The value of this expression was used to complete the "binary" column of Table 1.

vars	clauses	bounds and tableau		tableau only	
		binary	ternary	binary	ternary
75	322	51	48	258	248
100	430	54	45	368	341
150	645	61	59	1286	1084
200	860	55	51	12962	8632

Table 2: Time in seconds to answer 1000 random queries using the bounds together with the program tableau (a version of the Davis-Putnam algorithm) versus using tableau alone. All experiments were run on a 100Mz SGI Challenge workstation.

The probability that the bounds answer a random *ternary* query can be similarly derived, and was used to complete the final column of the table. As we have noted, a similar analysis could be performed for Horn bounds, but would be much more complicated, because the probability that a Horn theory entails a disjunction is the same as the probability that it entails one of the disjuncts.

As we can see from Table 1, the percentage of queries that can be handled by even such simple unit clause bounds is quite high. Note that the queries handled by the bounds can be answered in linear time. However, the Davis-Putnam procedure scales exponentially on the queries considered in the table (this follows from the experiments in [Mitchell *et al.*, 1992]). Thus, this suggests that knowledge compilation on such randomly-generated theories should have a clear payoff.

We verified the computational savings suggested by the preceding analysis by implementing the fast querying algorithm shown in Fig. 2, and testing 1000 random binary and 1000 random ternary queries against each of the 40 test theories.

In case both bounds failed to answer a query, it was tested against the original theory using an efficient implementation of the Davis-Putnam procedure called "tableau".[6] Table 2 lists the average time to run each batch of a 1000 queries, using the bounds together with tableau versus using tableau alone. Thus, in both cases *all* queries were answered. We see that knowledge compilation reduced

[6]The Davis-Putnam procedure is currently the fastest known complete procedure for propositional satisfiability testing and theorem-proving on the class of formulas considered here [Buro and Büning, 1992, Dubois *et al.*, 1995]. Tableau [Crawford and Auton, 1993] is one of the fastest implementations of the algorithm.

the overall time by over two orders of magnitude on the largest theories. This eliminates the remote possibility that the bounds are only answering the "easy" queries. Earlier we invoked complexity theory to argue that in *general* the bounds are not limited to easy queries; these experiments verify that the bounds answer hard queries against a computationally interesting distribution of random theories.

As an aside, we observe that even when we take into account the time required to compile the theories, we obtain an overall time savings. For example, on the 200 variable theories, computing the bounds takes about an hour and five minutes; thus, the total time to compute the bounds *and* answer 1000 binary queries is 3,955 seconds, versus 12,962 seconds not using the bounds. (Note that difference in overall time will increase even further when we would consider, for example, 10000 queries.) Thus in this case we have gone beyond our stated goal of shifting computational effort from on-line to off-line, and have actually reduced the total amount of work required.

These positive results for random theories are quite surprising, since one would expect that their apparent lack of structure would make them hard to approximate by simpler theories. A further understanding of the costs and benefits of the knowledge compilation approach will come through the application of the techniques to various real-world problems. Note that, in practice, one may be satisfied with upper and lower bounds that are not necessarily optimal, but are cheaper to compute. Greiner [1992] gives compilation algorithms that begin answering queries with non-optimal bounds, and gradually improve them at run-time depending on the particular query distribution.

2.5 Size of the Approximations

The size of the approximations is clearly an important issue. Ideally, one would like the approximations to be of roughly the same size or less than the original theory. If the approximations are much larger, then one may lose the potential computational advantage. For example, if a bound is exponentially larger than the original theory, evaluating a query in linear time using the bound would be as expensive as using the original theory.

From the **Generate_GLB** algorithm it follows immediately that the size of the generated bound is less than or equal to that of the original theory.[7] The system

[7] Recently, Cadoli [1993], building on our original paper on knowledge compilation [Selman and Kautz, 1991], has shown that computing the GLB is in the complexity class P^{NP}. (Problems

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." The LUB is less well-behaved. First, we will show that there are theories that have exponentially long upper-bounds. However, such bounds can often be shrunk down to the size of the original theory by introducing new letters in the vocabulary. Such new letters can be viewed as useful abbreviations for repeated structure in the theory. Finally, we will prove that, unfortunately, one cannot *always* shrink down the LUB by introducing new letters. In those cases, the system has to limit the length of the approximation by using a weaker bound. (We discuss some kinds of weaker bounds in Section 3; see also Greiner and Schuurmans [1992] for an example of this.)

2.5.1 Explosion of the LUB

The following theory has an LUB with has exponentially many clauses. The clauses in the theory can be interpreted as rules for deciding if someone is a cognitive scientist. The clauses are numbered for ease of reference.

$$\big(\text{CompSci} \wedge \text{Phil} \wedge \text{Psych}\big) \supset \text{CogSci} \tag{1}$$

$$\text{ReadsMcCarthy} \supset \big(\text{CompSci} \vee \text{CogSci}\big) \tag{2}$$

$$\text{ReadsDennett} \supset \big(\text{Phil} \vee \text{CogSci}\big) \tag{3}$$

$$\text{ReadsKosslyn} \supset \big(\text{Psych} \vee \text{CogSci}\big) \tag{4}$$

Clause (1) states a sufficient condition for being a cognitive scientist: being a computer scientist, *and* a philosopher, *and* a psychologist. The remaining clauses let one deduce a person's profession from his or her reading habits. Clause (2) states that if a person reads papers written by McCarthy, then the person is either a computer scientist or a cognitive scientist (or possibly both). Similarly, a reader of Dennett is either a philosopher or cognitive scientist or both, and a reader of Kosslyn is either a psychologist or cognitive scientist or both.

Reasoning with this theory can be quite complicated. For example, by reasoning by cases, one can prove that a computer scientist who reads Dennett and

in P^{NP} can be solved in polynomial time by a deterministic Turing machine with access to an NP oracle. See Garey and Johnson [1979] or Johnson [1990].) This means that computing a GLB is only slightly harder than answering a single query, since the best known algorithms for both problems are singly-exponential [Johnson, 1993]. Note that is indeed a positive result, because one might have feared that, in the worst case, a doubly-exponential amount of work was needed to compute a bound that can potentially handle an exponential number of queries efficiently.

Kosslyn is also cognitive scientist. In general, for such non-Horn form theories, finding a proof will take time exponential in the length of the theory (provided P$\neq$NP).

Clause (1) can be resolved with subsets of clauses (2—4) to yield many different Horn clauses, such as

$$\big(\text{ReadsMcCarthy} \wedge \text{Phil} \wedge \text{ReadsKosslyn}\big) \supset \text{CogSci}$$
$$\big(\text{CompSci} \wedge \text{ReadsDennett} \wedge \text{Psych}\big) \supset \text{CogSci}$$

In fact, the LUB of this theory is equivalent to the set of 2^3 Horn clauses:

$$\left\{ \begin{array}{l} (p \wedge q \wedge r) \supset \text{CogSci} \quad \big| \\ \qquad p \in \{\text{CompSci}, \text{ReadsMcCarthy}\} \\ \qquad q \in \{\text{Phil}, \ \text{ReadsDennett}\} \\ \qquad r \in \{\text{Psych}, \ \text{ReadsKosslyn}\} \end{array} \right\} \tag{5}$$

Furthermore, we can prove that there is *no* smaller set of clauses equivalent to (5). Note that this is a much stronger condition then simply saying that there are no redundant clauses in (5); we are asserting that there is no way to represent the same information in less space in clausal form. This example can be generalized by increasing the number of propositions that appear in the left-hand side of clause (1), and adding a clause of the general form of (2) for each new proposition. The size of the LUB then grows exponentially. In general, we have the following theorem.

Theorem 6 *There exist clausal theories Σ of size n such that the smallest clausal representation of their LUB is of size $O(2^n)$.*

The proof appears in the appendix. Therefore, although we can tell if any clause follows from the LUB in time linear in the size of the LUB, the explosion in size of the LUB in this example wipes out our savings. Of course, there are also many commonsense theories for which such exponential blowup does *not* occur.

2.5.2 Shrinking the LUB

The syntactic form of a theory can often be modified without changing its basic meaning. For example, any theory can be represented by a set of clauses each containing no more than three literals (3-CNF form) by introducing new propositional letters. The old and new theories are not equivalent, since the new one

uses an expanded vocabulary, but they are essentially the same: They both entail or both do not entail any formula that does not contain any of the new letters.

Thus one might wonder if a large LUB could be represented by a small set of Horn clauses that have basically the same meaning, if not actual logical equivalence. As with the case of 3-CNF formulas, the technique we use depends on the introduction of new propositional letters. Rather than modify the definition of the LUB, we will add these new letters to the source theory itself. If we take the meaning of a letter to be a concept, we will see that the method reduces to the definition of new concepts that generalize concepts in the original theory.

For example, let us modify the theory given by clauses (1—4) by introducing three new concepts, "computer science buff", "philosophy buff", and "psychology buff". The first generalizes the concepts of a computer scientist and of a reader of papers by McCarthy. Similarly, the second generalizes philosopher and reader of Dennett, and the third generalizes psychologist and reader of Kosslyn. Each concept definition requires three clauses: one to assert that the more general concept is divided among its subconcepts, and two to assert that the subconcepts are part of the concept. The added clauses are

$$\text{CompSciBuff} \supset (\text{CompSci} \vee \text{ReadsMcCarthy}) \tag{6}$$

$$\text{CompSci} \supset \text{CompSciBuff} \tag{7}$$

$$\text{ReadsMcCarthy} \supset \text{CompSciBuff} \tag{8}$$

$$\text{PhilBuff} \supset (\text{Phil} \vee \text{ReadsDennett}) \tag{9}$$

$$\text{Phil} \supset \text{PhilBuff} \tag{10}$$

$$\text{ReadsDennett} \supset \text{PhilBuff} \tag{11}$$

$$\text{PsychBuff} \supset (\text{Psych} \vee \text{ReadsKosslyn}) \tag{12}$$

$$\text{Psych} \supset \text{PsychBuff} \tag{13}$$

$$\text{ReadsKosslyn} \supset \text{PsychBuff} \tag{14}$$

The LUB of the augmented theory containing (1—4) and the clauses above can be represented by just the Horn clauses from the new concept definitions (7, 8, 10, 11, 13, 14) together with the single clause

$$(\text{CompSciBuff} \wedge \text{PhilBuff} \wedge \text{PsychBuff}) \supset \text{CogSci} \tag{15}$$

The total size of the LUB (counting all occurrences of literals) is half the size of the LUB of the unmodified theory. As we generalize the example by adding new

propositions, the size of the LUB grows as $O(n)$, as opposed to $O(2^n)$. Furthermore, the added clauses only increase the size of the source theory by a linear amount.

Returning to our example concerning a computer scientist who reads Dennett and Kosslyn, we can now infer quickly from (7), (11), and (14) that the person is a computer science buff, philosophy buff, and psychology buff, and therefore by (15) a cognitive scientist. Note that this inference can be computed in time linear in the size of the new LUB and therefore linear in the size of the *original* theory (1—4). Thus, by teaching the system new concept definitions, the size of the new source theory grows only slightly, and the LUB shrinks to approximately the size of the source theory itself.

In [Kautz and Selman, 1992a] we discuss a strategy for *automatically* deriving ("learning") new concept letters that shrink the original bound. We call this *theory compaction.* The flavor of our approach is very similar to that of the use of additional letters in extended resolution [Tseitin, 1968]. See [Muggleton and Buntine, 1988] for a related approach to learning new generalizations, based on inverting resolution proofs.

2.5.3 Does a Compact Representation of the LUB Always Exist?

So far, we have shown that a naive representation of a theory's LUB can sometimes require an exponential amount of space, and that in some of those cases a clever representation using new propositional letters requires only a polynomial amount of space. The question then becomes how general these results are. Is it *always* possible to produce a compact, tractable representation of the LUB? Unfortunately, the following theorem shows that this is unlikely.

Theorem 7 *Unless NP $\subseteq$ non-uniform P, it is not the case that the Horn least upper-bound $\Sigma_{\rm lub}$ of a propositional clausal theory Σ can always be represented by a data structure that allows queries of the form*

$$\Sigma_{\rm lub} \models \alpha$$

to be answered in time polynominal in $(|\Sigma|+|\alpha|)$, where α is a single Horn clause.

Note that this is so despite the fact that we allow an arbitrary amount of work to be performed in computing the data structure used to represent $\Sigma_{\rm lub}$. The proof of the theorem appears in the appendix.

The notion of "non-uniform P" comes from work in circuit complexity [Boppana and Sipser, 1990]. A problem is in non-uniform P (also called P/poly) iff for every integer n there exists a circuit of complexity (size) polynomial in n that solves instances of size n. The adjective "non-uniform" refers to the fact that different circuits may be required for different values of n. Any problem that can by solved by an algorithm that runs in time $O(f(n))$ has circuit complexity $O(f(n)\log n)$. We use this fact implicitly in the proof of the theorem, where we talk about polynomial time algorithms rather than polynomial size circuits.

The class non-uniform P is, however, considerably larger than P. (For example, non-uniform P contains non-computable functions, such as the function that returns "1" on inputs of length n iff Turing Machine number n halts on all inputs. For any n the circuit is simply fixed to return 1 or 0.) Although it is possible that P $\neq$ NP and yet NP $\subseteq$ non-uniform P, this is considered unlikely. One consequence would be that the polynomial-time hierarchy would collapse to Σ_2 [Karp and Lipton, 1982]. As shown in the appendix, the theorem can be strengthened to say that the claim that there always exists an efficient form of the LUB for answering Horn clausal queries is *equivalent* to the claim that NP $\subseteq$ non-uniform P. Therefore a proof that such efficient representations do or do not always exist would be a major result in the complexity of finite functions.

An immediate corollary of the theorem is that unless the polynomial hierarchy collapses in this manner, compaction by defining new propositions is an incomplete method for shrinking the LUB.

Corollary *Unless NP $\subseteq$ non-uniform P, it is not the case that there is always a compaction (using any number of new variables) of the Horn least upper-bound of a theory Σ that is of size polynomial in $|\Sigma|$.*

This follows from the theorem because one can determine if a Horn clause follows from a compaction (which is itself Horn) in time linear in the length of the compaction plus the length of the query.

To summarize, there are theories for which the Horn LUB cannot be represented in a short, tractable form. In such cases, knowledge compilation can still be of use, but one will have to rely on a shorter and therefore weaker upper-bound. One way to do this would be to simply stop generating the bounds at some point; this is straightforward, given the anytime nature of our compilation algorithms. Another alternative would be to generate so-called k-Horn approximations, which are Horn theories with at most k literals per clause [Dechter and Pearl, 1992]. Given the limitation on the clause lengths, such theories are of most polynomial

size in the number of variables. (We discuss k-Horn and other alternatives to Horn approximations in Section 3 below.)

3 General framework

Central to our knowledge compilation proposal is the definition of approximations in terms of subsets and supersets of the set of models of the original theory. This was illustrated in Fig. 1. Such a model-theoretic definition leaves much freedom in terms of choice of representation language for the original theory, the approximations, and the queries. Our only additional requirement is that the queries can be evaluated efficiently with respect to the approximations. Note again that we view the computation of the approximations as an off-line compilation process, so that its cost can be amortized over the set of queries. By allowing substantial computational resources to be spend on the compilation the approximations will become of real practical significance, as was shown in Section 2.4.

Formally, a *knowledge compilation system* is a tuple $\langle \mathcal{L}, \models, \mathcal{L}_{\rm S}, \mathcal{L}_{\rm T}, \mathcal{L}_{\rm Q}, f_{\rm L}, f_{\rm U} \rangle$ containing the following components:

- $\mathcal{L}$ is a formal language. We identify a language with the set of all its sentences.
- $\models$ is a consequence relation over sets of sentences in $\mathcal{L}$. In most of the examples we will study, $\models$ has its usual meaning of logical entailment, but the framework allows $\models$ to represent other relationships, such as subsumption.
- $\mathcal{L}_{\rm S}$ is the "source" sublanguage of $\mathcal{L}$, used to express a general knowledge base.
- $\mathcal{L}_{\rm T}$ is the "target" sublanguage of $\mathcal{L}$, used to express approximations to the general knowledge base. It should be easier in some sense (analytically or empirically) to determine if a query is a consequence of a set of sentences in $\mathcal{L}_{\rm T}$ than of a set sentences in $\mathcal{L}_{\rm S}$.
- $\mathcal{L}_{\rm Q}$ is the "query" sublanguage of $\mathcal{L}$.
- $f_{\rm L}$ is a function that models some (particular) anytime lower-bound algorithm, mapping a source theory and time-bound n to the

best lower-bound that can be computed in n steps. Formally, $f_{\mathrm{L}} : \mathcal{L}_{\mathrm{S}} \times \mathrm{N} \rightarrow \mathcal{L}_{\mathrm{T}}$. By definition, $f_{\mathrm{L}}(\Sigma, \infty)$ is a GLB of Σ.

f_{U} is a similar upper-bound function, where $f_{\mathrm{U}}(\Sigma, \infty)$ is the LUB of Σ.

Suppose a knowledge compilation system is presented a query $\alpha \in \mathcal{L}_{\mathrm{Q}}$ after performing i compilation steps on the source theory Σ. As described earlier, if $f_{\mathrm{L}}(\Sigma, i) \not\models \alpha$ then the system answers "no", if $f_{\mathrm{U}}(\Sigma, i) \models \alpha$ then the system answers "yes", and otherwise it answers "unknown". A definite answer of "yes" or "no" always agrees with the answer to the question "Does $\Sigma \models \alpha$?", even if the lower (upper) bound is not the greatest (least) bound.

The case of knowledge compilation using propositional Horn clauses fits into this framework as follows: $\mathcal{L}$, $\mathcal{L}_{\mathrm{S}}$, and $\mathcal{L}_{\mathrm{Q}}$ are general, propositional, clausal languages, $\mathcal{L}_{\mathrm{T}}$ is restricted to Horn clauses, and $\models$ is propositional entailment. In this case the query language is strictly more expressive than $\mathcal{L}_{\mathrm{T}}$. (Note that to answer a general query consisting of a set of clauses, one can determine efficiently whether each of the clauses is implied by the Horn approximation, by using the linear time algorithm for testing the satisfiability of Horn clauses.) The functions f_{L} and f_{U} model the **Generate_GLB** and **Generate_LUB** algorithms presented above.

3.1 Other Instances of Knowledge Compilation

3.1.1 Restricted Propositional Clausal Forms

A broad class of propositional knowledge compilation systems can be realized by generalizing the algorithms for the Horn case. The idea of a Horn-strengthening is generalized as follows:

Definition: θ-strengthening
Let θ be a particular class of propositional clauses. A clause in that class is called a θ-clause, and a set of such clauses is called a θ-theory. A clause C_θ is a θ-strengthening of a clause C iff $C_\theta \subseteq C$ and there is no θ-clause C'_θ such that $C_\theta \subset C'_\theta \subseteq C$.

A knowledge compilation system can be created by letting the source language $\mathcal{L}_{\mathrm{S}}$ be the set of general clauses, and the target language $\mathcal{L}_{\mathrm{T}}$ be such a restricted class of clauses θ. Now further suppose that θ is closed under resolution (*i.e.*, the resolvent of any two θ-clauses is a θ-clause), and that every clause has at least one

θ-strengthening. Then we see that the proof of Lemma 1 generalizes from Horn clauses to θ-clauses: that is, if a θ-theory entails a clause C, then it must also entail a θ-strengthening of C. Furthermore, every greatest lower-bound in terms of θ-clauses of a theory is equivalent to a (logically weakest) θ-strengthening of the theory. Therefore one can find a GLB by searching the space of θ-strengthenings.

Theorem 8 *Let θ be a class of propositional clauses such that (i) the resolvent of any two clauses in θ is also in θ, and (ii) every clause is subsumed by some clause in θ. Given a set of clauses Σ,* **Generate_GLB** *(Fig. 3) computes a greatest θ-lower-bound of Σ of length less than or equal to that of Σ (where "θ" is substituted for "Horn" in the algorithm).*

Now consider the problem of generating the least upper-bound of a clausal theory. Recall that the "naive" algorithm for finding the Horn LUB of a theory is simply to compute all resolvents of the theory, and then collect the Horn prime implicates from that set. This is due to the resolution completeness theorem and the fact that propositional Horn clauses are closed under *subsumption*, *i.e.*, a Horn clause can only be subsumed by another Horn clause. Therefore, a condition (i) we place on the general target language θ is that it be closed under subsumption. (An example of a class that violates this condition is the set of definite Horn clauses. These are clauses that contain exactly one positive literal. The resolvent of any two definite Horn clauses is a definite Horn clause. However, a definite Horn clause (for example, $\neg p \vee q$) can be subsumed by a non-definite Horn clause (for example, $\neg p$)). We summarize these observations about generating the θ-LUB in the following theorem:[8]

Theorem 9 *Let θ a class of propositional clauses such that that if $C \in \theta$ and C' subsumes C, then $C' \in \theta$. Given a set of clauses Σ,* **Generate_LUB** *(Fig. 4) computes the least θ-upper-bound of Σ (where "θ" is substituted for "Horn" in the algorithm, and C_0 is selected from $\Sigma_\theta \cup \Sigma_{\rm N}$).*

Several useful clausal target languages meet the conditions of Theorems 8 and 9. These include:

- Horn and Reverse-Horn (clauses containing at most one *negative* literal).

[8] This theorem is due to del Val [1995], and strengthens a proposition from Kautz and Selman [1991] that appeared in earlier drafts of this paper.

- Binary: clauses containing two or fewer literals. Satisfiability and entailment can be determined in linear time for this class [Aspvall *et al.*, 1979]. (Note that unit clauses and the empty clause are included in this class.)

- Unit clauses; that is, bounds that consist of conjunctions of literals. This class is a specialization of both Horn and binary.

- Clauses *not* containing a given set of propositional letters. While such a class may not have better worst-case complexity than the unrestricted language, it may be empirically desirable to "compile away" certain irrelevant propositions.[9] For example, let $\Sigma = \{p \vee q,\ \ \neg q \vee r\}$, and θ be clauses not containing q. Then the θ-GLB of Σ is $\{p \wedge r\}$, and θ-LUB is $\{p \vee r\}$.

There are still other interesting tractable clausal target languages that require more significant modifications to our compilation algorithms. Important ones include:

- k-Horn: for any fixed k, Horn clauses containing k or fewer literals. Note that reasoning with k-Horn theories takes only linear time, and furthermore, the size of the k-Horn approximations is always polynomial in the number of variables. One way to compute a k-Horn GLB is to simply generate the set of all Horn clauses of length k as an initial (inconsistent) lower bound, and then minimizing the set by striking out clauses until it no longer entails the source theory. There is a similar brute-force algorithm to compute the k-Horn LUB: start with the set of all k-length clauses entailed by the source theory, and then repeatedly shrink it by removing clauses that are entailed by the conjunction of the remaining clauses.

- Renamable Horn: the class of theories that become Horn by uniformly replacing a subset of the variables by their negation. This class is of particular interest because (linear-time) unit propagation (BCP) is complete for it [Henschen and Wos, 1974, Lewis, 1978].

In additional to the clausal target languages we have concentrated on in this paper, one could consider tractable *non*-clausal target languages. For example,

[9] Subramanian and Genesereth [1987] present a formal system for inferring that certain propositions are irrelevant to the computation of a given class of queries. Given this sense of what is irrelevant, knowledge compilation can then be used as way to simplify the theory.

McAllester and Given [1992] discuss a logical language based on the structure of natural language, and identify a fragment in which inference can be performed in polynomial time. It would be interesting to see if ordinary first-order theories could be (approximately) compiled into this tractable fragment, using our model-theoretic notion of approximation.

3.1.2 First-Order Theories

We will now consider the application of knowledge compilation in first-order languages. Here will we only give a brief description of how to extend knowledge compilation techniques to languages with quantification; more details will appear in a future paper [Kautz and Selman, forthcoming].

In the propositional case, GLBs and LUBs for any pair of a source and target language always exist, since only a finite number of logically distinct theories can be constructed. In the general first-order case, however, such best approximations may not be well-defined. For a simple example, let $\mathcal{L}_{\mathrm{S}}$ be first-order logic, with a single predicate p and the infinite set of constant terms $\{c_1, c_2, \ldots\}$, and $\mathcal{L}_{\mathrm{T}}$ consist of ground (non-quantified) clauses. The theory $\{\exists x.p(x)\}$ has an infinite series of better and better ground-clause lower-bounds:

$$\{p(c_1)\} \models \{p(c_1) \vee p(c_2)\} \models \{p(c_1) \vee p(c_2) \vee p(c_3)\} \models \cdots \models \{\exists x.p(x)\}$$

That is, there is no one *greatest* lower-bound.

The notion of a GLB is well-defined, however, for the first-order generalizations of the *clausal* languages we have considered. (In fact, clausal representations are the most widely used first-order languages.) A *first-order literal* is constructed in the usual way from an optionally negated predicate symbol applied to a sequence of terms constructed from function symbols, constants, and variables. A *first-order clause* is a sentence in prenex form containing only universal quantifiers, whose matrix is a disjunction of literals.

A first-order Horn clause is a first-order clause whose matrix contains at most one positive literal; a Horn-strengthening of a first-order clause corresponds to a Horn-strengthening of its matrix. Note that even in the first-order case, a clause has only a *finite* number of Horn-strengthenings.

The reader may now verify that Lemmas 1 and 2 hold without change in the first-order case. Let us supply **Generate_GLB** in Fig. 3 with an oracle for the test

if $L \models L'$ **then** $L := L'$

Then it clear that given a set of first-order clauses, the algorithm will search through a finite set of Horn-strengthenings and return one that is logically weakest; by the argument in the proof of Theorem 2, this must be a GLB. This shows that first-order Horn GLBs exist and are small.

There is also no difficulty in generalizing the notion of a GLB in other restricted clausal target languages to the first-order case. The θ-strengthening of a first-order clause is again generated by eliminating literals from the clause's matrix. Then, Theorem 8 goes through as before.

It is less straightforward to apply the notion of a LUB to first-order theories. We can construct finite first-order clausal theories that have no finite Horn LUB (even using no function symbols, only variables). Thus, not all first-order Horn LUBs are finitely representable. There are other aspects of compiling first-order theories that will be dealt with in our later paper. In particular, we are investigating decidable (and possibly tractable) approximations to theories in undecidable languages.[10]

3.1.3 Description Logics

In this section, we consider description logics, a family of frame-based knowledge representation languages as studied by Levesque and Brachman [1985]. (See also [Donini *et al.*, 1991].)

Levesque and Brachman consider a language $\mathcal{FL}$ in which one can describe structured concepts in terms of other concepts, either complex or primitive. For example, if we wished to describe the concept of a person whose male friends are all doctors with some specialty, we could use the $\mathcal{FL}$ expression

```
(AND person
     (ALL (RESTR friend male)
          (AND doctor
               (SOME specialty)))),
```

which contains all the constructs (the capitalized terms) used in the language. Levesque and Brachman consider the complexity of determining whether one con-

[10] Interesting connections may be drawn with some recent research on the analysis of Prolog programs (for use in, for example, optimization and program specification). For example, [Heintze and Jaffar, 1990] describes how to construct a *recursive* (*i.e.*, decidable) approximation to a potentially non-recursive logic program. Their method computes a lower-bound of the logic program viewed as a logical theory (but not in general the greatest lower-bound).

cept *subsumes* another. For example, the concept of "a person whose friends are all doctors", written

```
(AND person
     (ALL friend
          doctor))
```

subsumes the one given above. Now, their central technical result is that determining subsumption in $\mathcal{FL}$ is intractable, but that removing the RESTR construct leads to polynomial time computable subsumption. The restricted language is called $\mathcal{FL}^-$.

So for efficient subsumption, one can use the language $\mathcal{FL}^-$. But this language may not be sufficiently expressive for practical applications. Knowledge compilation provides again an alternative. In this case the idea is to take a concept description in the language $\mathcal{FL}$ and to approximate it using two concept descriptions in $\mathcal{FL}^-$: a best lower-bound, *i.e.*, the most general more specific concept in $\mathcal{FL}^-$, and a best upper-bound, *i.e*, the most specific more general (subsuming) concept in $\mathcal{FL}^-$.

As an example consider the first concept given above. It is not difficult to see that the concepts person and

```
(AND person
     (ALL friend
          (AND doctor
               (SOME specialty))))
```

in $\mathcal{FL}^-$ are examples of, respectively, an upper-bound and a lower-bound in $\mathcal{FL}^-$. (These are also the best bounds in this case.) A knowledge representation system can store such bounds with the original concept description, and use them to try to determine quickly whether the newly given concept subsumes it or is subsumed by it.

Note that in this application we compute a pair of bounds for each concept, instead of a single pair of bounds for an entire knowledge base. The bounds can be used both in subsumption queries to the KB and in updating the KB by adding new concept definitions. In both cases, the system can use the bounds in determining the appropriate place of the new concept in the hierarchy.

4 Related Work

An upper-bound approximation is logically weaker than the original theory, and can thus be viewed as a *generalization* or *abstraction* of the theory. A lower-bound approximation is a *specialization* (logical strengthening) of the original theory. Borgida and Etherington [1989] 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

$$\texttt{doctor(Jill)} \supset \texttt{professional(Jill)}$$
$$\texttt{lawyer(Jill)} \supset \texttt{professional(Jill)}$$

and the KB is

$$\texttt{doctor(Jill)} \vee \texttt{laywer(Jill)}.$$

They then generate a new KB that contains only $\texttt{professional(Jill)}$. 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 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. (Related to the work of Borgida and Etherington is that of Imielinski [1987], who proposes a notion of abstraction based on grouping individuals into equivalence classes.)

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 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.

Horn lower-bounds are also a generalization of Levesque's notion of vivid representations [Levesque, 1986, Etherington *et al.*, 1989]. To allow for fast inference, a vivid representation contains only complete information (no "reasoning by cases"). Levesque proposes to replace a knowledge base containing incomplete information by a complete, vivid representation of the information. Such a representation can be generated using abstractions or by filling in details. In the

propositional case, the proposal comes down to replacing a propositional theory by one of its models (which is presumably "prototypical" of the models of the original theory). Queries are evaluated directly with respect to the model. Our Horn lower-bound can be viewed as a relaxation of the completeness constraint: Instead of picking a single model, we use a collection of models represented by a set of Horn clauses. We maintain the computational advantage because inference is still efficient using the Horn bound.

In work in artificial intelligence on planning and theorem-proving with abstraction [Amarel, 1968, Plaisted, 1981], one maps a theory to a smaller, simpler theory, generates proofs in the smaller theory, and then uses the proofs to guide generation of proofs in the original theory. Various mechanisms have been suggested for creating the abstractions, such as eliminating operator preconditions [Sacerdoti, 1974], finding symmetries in the search space [Ellman, 1993], and employing explanation-based learning on previously-generated plans [Knoblock *et al.*, 1991a]. These abstractions are similar to our notion of an upper-bound. (Some of the approaches differ in that they can introduce abstract solution that do not correspond to any concrete solution. Knoblock *et al.* [1991] discuss various kinds of abstractions that do preserve such solution and consistency properties.)

A crucial difference between the work on planning and our knowledge compilation framework is that the former concentrates on solving single problem instances, rather than shifting computational effort from on-line to off-line. All but the most restricted forms of planning are NP-hard [Gupta and Nau, 1991, Bylander, 1991, Erol *et al.*, 1992]. Therefore, the best known planning algorithms run in exponential time. Work on planning with abstraction attempts to lower the exponent. By contrast, the aim in the knowledge compilation framework is to preprocess (compile) a theory in such a manner that queries (problem instances) can be solved in polynomial time.

The motivation behind our knowledge compilation approach is similar to that behind the work on *universal plans* [Schoppers, 1987], a form of reactive planning [Agre and Chapman, 1987]. A universal plan is a program (or logical circuit) that efficiently determines the first step of a plan leading from any initial state toward a pre-defined goal state. (After this step is taken, the universal plan determines what to do in the new state, and so on, until the goal is reached.) Universal plans can thus be viewed as an attempt to efficiently represent a potentially exponential set of plans, just as knowledge compilation aims at creating a representation that can efficiently answer a potentially exponential set of queries. In universal plans, one generally tries to capture exactly the original domain theory. It would be

interesting to explore the question of whether Horn approximations of the domain knowledge could be used to create "approximate" universal plans (*e.g.*, ones that may not specify for *every* state what to do next).

Cadoli and Schaerf [1992] generalize Levesque's work on implicit and explicit belief [Levesque, 1984]. They approximate the inference process by allowing a sequence of more and more powerful inference relations. The more time the system has to evaluate a query, the more powerful an inference relation it can use. Allowing for an unrestricted amount of time leads to standard deductive reasoning. An important difference with our approach is that there is no notion of directly processing the original theory to obtain an approximation to it. This means that in practice only a limited amount of work can be done per query, in order to obtain an answer. Moreover, none of the work done for one query is saved for use in answering the next query. In our framework, the cost of run-time querying is shifted to off-time compilation. The cost of the compilation process is amortized over a potentially very large set of queries, and the result of the compilation can be used in processing each query.

Dalal and Etherington [1992] provide a general framework in which many different forms of approximations can be captured. Our notion of an upper bound corresponds to their notion of a "direct weaking" of a theory. However, their notion of the strengthening of a theory is quite different from ours; it roughly corresponds to the disjunction of all of the lower-bounds of a theory. They introduce approximations to queries as well as theories; in our our framework there is no need to approximate queries — as we saw earlier, it is possible to test arbitrary CNF queries against Horn theories in linear time.

Dalal and Etherington do not develop the notion of shifting computation effort from query-answering time to an off-line compilation phase, nor do they present algorithms for compiling specific languages. The emphasis in their work is more on developing a very general framework for describing mappings between different vocabularies, rather than dealing with specific issues of computational complexity.

Dechter and Pearl [1992] investigate the use of Horn approximations to obtain more efficient representations of empirical data. They consider the case where the original theory Σ is given directly by its set of models (satisfying truth assignments). They introduce a weaker notion of Horn approximations, called k-Horn approximations, in which each Horn clause contains at most k literals. See [Kautz *et al.*, 1994], for a discussion of this work.

Finally, Greiner and Schuurmans [1992; Greiner 1992] have adapted our com-

pilation algorithms to ones that search for bounds that are optimal with respect to a given query distribution. Generation of the bounds is interleaved with query answering, and the queries themselves are used to optimize the bounds. They show that this can be done efficiently, in that each interation takes polynomial time.

5 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 (tractable) language. A representation system incorporating a knowledge compilation procedure will allow the user to enter information in a general, unrestricted language, which then the system compiles 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.

We also presented an empirical evaluation of our approach, and demonstrated substantial computational savings on hard propositional theories.

We showed how the procedures for compiling propositional theories into Horn theories can be generalized to apply to other tractable classes of clauses. Those classes were characterized using various closure conditions. The classes containing reverse-Horn clauses, clauses with two or fewer literals, or clauses not containing a certain set of "irrelevant letters" are examples of classes that satisfy the closure conditions. Finally, we discussed the compilation of concept descriptions given in a terminological representation language. This example showed that our knowledge compilation approach appears suited for dealing with a variety of knowledge representation languages — not only traditional logics.

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, Yoav Shoham, and two anonymous referees for many useful comments.

Appendix: Proofs

Proof of Theorem 6

Consider a theory Σ of the following form, for arbitrary n:

$$\neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_n \vee s \tag{16}$$
$$\neg q_1 \vee p_1 \vee s \tag{17}$$
$$\neg q_2 \vee p_2 \vee s \tag{18}$$
$$\vdots \tag{19}$$
$$\neg q_n \vee p_n \vee s \tag{20}$$

We see that Σ contains $4n + 1$ literals; that is, it is of size $O(n)$. The LUB Σ_{lub} of Σ is the following set of Horn clauses, of total length 2^n:

$$\left\{ \begin{array}{c} \neg x_1 \vee \neg x_2 \vee \cdots \vee \neg x_n \vee s \;\; | \\ x_i \in \{p_i, q_i\} \;\; \text{for} \;\; 1 \leq j \leq n \end{array} \right\}$$

First we prove that the set Σ_{lub} has the following properties: no two clauses resolve, and it is irredundant (no subset of Σ_{lub} implies all of Σ_{lub}). Then we prove

that $\Sigma_{\rm lub}$ is of minimum size. (Note that being of minimum size is a stronger condition than being irredundant.)

Proof that $\Sigma_{\rm lub}$ is irredundant: suppose there is a clause α in $\Sigma_{\rm lub}$ such that $\Sigma_{\rm lub} - \{\alpha\} \models \alpha$. Since no two clauses in $\Sigma_{\rm lub} - \{\alpha\}$ resolve, by completeness of resolution there must be an α' in $\Sigma_{\rm lub} - \{\alpha\}$ such that α' subsumes α. But this is impossible, since all clauses in $\Sigma_{\rm lub}$ are of the same length.

We can now prove that there is no smaller set of clauses $\Sigma'_{\rm lub}$ which is equivalent to $\Sigma_{\rm lub}$: Suppose there *were* a $\Sigma'_{\rm lub}$ such that $\Sigma_{\rm lub} \equiv \Sigma'_{\rm lub}$ and $|\Sigma'_{\rm lub}| < |\Sigma_{\rm lub}|$. Then for all α in $\Sigma'_{\rm lub}$,

$$\Sigma_{\rm lub} \models \alpha$$

and since no clauses in $\Sigma_{\rm lub}$ resolve, this means that there exists an α' in $\Sigma_{\rm lub}$ such that α' subsumes α.

That is, every clause in $\Sigma'_{\rm lub}$ is subsumed by some clause in $\Sigma_{\rm lub}$. Suppose each clause in $\Sigma_{\rm lub}$ subsumed a different clause in $\Sigma'_{\rm lub}$; then $|\Sigma_{\rm lub}{}'| \geq |\Sigma_{\rm lub}|$, a contradiction. Therefore there is a proper subset $\Sigma_{\rm lub}{}''$ of $\Sigma_{\rm lub}$ such that each clause in $\Sigma'_{\rm lub}$ is subsumed by some clause in $\Sigma_{\rm lub}{}''$.

Then $\Sigma_{\rm lub}{}'' \models \Sigma_{\rm lub}{}'$, and therefore $\Sigma_{\rm lub}{}'' \models \Sigma_{\rm lub}$. But this is impossible, because we saw that $\Sigma_{\rm lub}$ is irredundant. Therefore there is no smaller set of clauses equivalent to $\Sigma_{\rm lub}$ which is shorter than $\Sigma_{\rm lub}$. □

Proof of Theorem 7

Suppose such a representation of the LUB always existed. We then show that 3-SAT over n variables can be determined in $O(n^3)$ time. Because the choice of n is arbitrary, and 3-SAT is an NP-complete problem, this would mean that NP $\subseteq$ non-uniform P.

Let the variables be a set of main variables $\{p_1, \ldots, p_n\}$ together with a set of auxiliary variables

$$\{c_{x,y,z} | x, y, z \in \text{LITS}\}$$

where LITS is the set of literals constructed from the main variables (the variables or their negations). Let the source theory Σ be the conjunction of clauses:

$$\Sigma = \bigwedge_{x,y,z \in \text{LITS}} \{x \vee y \vee z \vee \neg c_{x,y,z}\}$$

Note that Σ is of length $O(n^3)$. The idea behind the construction is that any 3-SAT clause over n variables can be constructed by selecting a subset of the clauses from Σ and eliminating the auxiliary variables.

Now suppose we have been able to compute a representation of the Horn LUB of Σ that has the property that one can test Horn queries against it in polynomial time. We noted before that a Horn formula follows from the LUB if and only if it follows from the source theory.

Suppose Δ is an arbitrary 3-CNF formula over n variables that we wish to test for satisfiability. We construct a Horn clause α containing only auxiliary variables by including a negative auxiliary variable c that corresponds to each clause δ in Δ. For example, if Δ is

$$(p_1 \vee \neg p_3 \vee p_5) \wedge (\neg p_1 \vee p_2 \vee \neg p_4)$$

then the corresponding Horn clause is

$$\neg c_{p_1,\neg p_3,p_5} \vee \neg c_{\neg p_1,p_2,\neg p_4}$$

Now we claim that this Horn clause is implied by the LUB if and only if Δ is not satisfiable.

($\rightarrow$) Suppose the query is implied by the LUB. It follows that

$$\Sigma \models \neg c \vee \neg c' \vee \neg c'' \vee \ldots$$

where the $\{c, c', c'', \ldots\}$ are the auxiliary variables in the query that correspond to clauses $\{\delta, \delta', \delta'', \ldots\}$ in Δ. Equivalently,

$$\Sigma \cup \{\neg(\neg c \vee \neg c' \vee \neg c'' \vee \ldots)\} \text{ is unsatisfiable.}$$

That is,

$$\Sigma \cup \{c, c', c'', \ldots\} \text{ is unsatisfiable.}$$

Note that any clause in Σ containing an auxiliary variable other than $\{c, c' c'' \ldots\}$ can be eliminated, since that clause is satisfied in any model in which its auxiliary variable is assigned false, and no other instance of that variable appears in the formula. Thus it must be the case that

$$\{\delta \vee \neg c, \delta' \vee \neg c', \delta'' \vee \neg c'', \ldots\} \cup \{c, c', c'', \ldots\}$$

is unsatisfiable. Because the auxiliary variables each appear exactly once negatively and once positively above, they can can be resolved away. Therefore

$$\{\delta, \delta', \delta'', \ldots\} \equiv \Delta \text{ is unsatisfiable.}$$

($\leftarrow$) Note that each step in the previous section can be reversed, to go from the assumption that Δ is unsatisfiable to the conclusion that $\Sigma \models \alpha$. And, since α is Horn, it follows that the LUB implies α.

We assumed that the test whether the LUB implies α could be performed in time polynominal in the length of the source theory plus the length of the query. Since both the source theory and query are polynomial in the length of Δ, it follows that the satisfiability of Δ can be determined in time polynomial in the length of Δ. Since the choice of n was arbitrary, and 3-SAT is an NP-complete problem, this means that NP $\subseteq$ non-uniform P. $\Box$

Proof of strengthened Theorem 7

We can strengthen the theorem to an equivalence by showing that NP $\subseteq$ non-uniform P implies that small and tractable representations of the LUB always exist. Suppose we are given a source theory Σ of length m containing n variables. Assuming NP $\subseteq$ non-uniform P, there exists a circuit that determines satisfiability of formulas of length $m + n$ that has complexity polynomial in $m + n$. We use this circuit to construct program to test queries of the form $\Sigma_{\text{lub}} \models \alpha$ as follows: given α, first check that is not a tautology, and eliminate any duplicated literals. The resulting query is of length $\leq n$. Then pad out the query to exactly length n by duplicating any of its literals. Then the negation of the query together with Σ is a formula of exactly length $m + n$, so we can use the circuit to determine if the formula is unsatisfiable, or equivalently, that α follows from Σ. Since α is Horn, then the latter condition is equivalent to saying $\Sigma_{\text{lub}} \models \alpha$. Since the circuit is of size polynomial in $m + n$ it must execute in time polynomial in $m + n$ — that is, in time polynomial in $(|\Sigma| + |\alpha|)$. $\Box$

References

[Agre and Chapman, 1987] P.E. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of AAAI-87*, pages 268–272, Seattle, Wa, 1987.

[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.

[Aspvall *et al.*, 1979] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulae. *Information Processing Letters*, 8(121), 1979.

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

[Boppana and Sipser, 1990] R. B. Boppana and M. Sipser. The complexity of finite functions. In J. an Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Compexity*, pages 757–804. Elsevier, Amsterdam (and MIT Press, Cambridge), 1990.

[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.

[Buro and Büning, 1992] M. Buro and H. Kleine Büning. Report on a sat competition. Technical Memorandum 110, Mathematik/Informatik Universität Paderborn, November 1992.

[Bylander, 1991] T. Bylander. Complexity results for planning. In *Proceedings of IJCAI-91*, pages 274–279, Sidney, Australia, 1991.

[Cadoli and Schaerf, 1992] Marco Cadoli and Marco Schaerf. Approximation in concept description languages. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR-92)*, pages 330–341, Cambridge, MA, 1992.

[Cadoli, 1993] Marco Cadoli. Semantical and computational aspects of horn approximations. In *Proceedings of IJCAI-93*, pages 39–44, Chambery, France, 1993.

[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.

[Crawford and Auton, 1993] J. Crawford and L. Auton. Experimental results on the crossover point in satisfiability problems. In *Proceedings of AAAI-93*, pages 21–27, Washington, DC, 1993.

[Dalal and Etherington, 1992] Mukesh Dalal and David W. Etherington. Tractable approximate deduction using limited vocabularies. In *Proceedings of the Ninth Canadian Conference on Artificial Intelligence (AI '92)*, pages 206–212, Vancouver, British Columbia, 1992.

[Davis and Putnam, 1960] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the Association for Computing Machinery*, 7:201–215, 1960.

[de Kleer, 1990] Johan de Kleer. Exploiting locality in the tms. In *Proceedings of AAAI-91*, pages 264–271, Anaheim, CA, 1990.

[Dean, 1985] Tom Dean. *Artificial Intelligence, Theory and Practice*. Benjamin/Cummings, Redwood City, CA, 1985.

[Dechter and Pearl, 1992] Rina Dechter and Judea Pearl. Structure identification in relational data. *Artificial Intelligence*, 58(1–3):237–270, 1992.

[Dechter, 1992] Rina Dechter. Constraint networks. In *Encyclopedia of Artificial Intelligence*, pages 276–285. John Wiley & Sons, New York, 1992.

[del Val, 1985] A. del Val. An analysis of approximate knowledge compilation. In *Proceedings of IJCAI-95*, Montreal, Canada, 1985.

[Donini *et al.*, 1991] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept languages. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR-91)*, pages 151–162, Cambridge, MA, 1991.

[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.

[Dubois *et al.*, 1995] O. Dubois, P. Andre, Y. Boufkhad, and J. Carlier. Sat versus unsat. In David S. Johnson and Michael A. Trick, editors, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science. AMS Press, Providence, RI, 1995.

[Ellman, 1993] Thomas Ellman. Abstraction via approximate symmetry. In *Proceedings of IJCAI-93*, volume 2, pages 916–921, Chambery, France, 1993.

[Erol *et al.*, 1992] K. Erol, D.S. Nau, and V.S. Subrahmanian. On the complexity of domain-independent planning. In *Proceedings AAAI-92*, pages 381–386, San Jose, CA, 1992.

[Etherington *et al.*, 1989] David W. Etherington, Alex Borgida, Ronald J. Brachman, and Henry Kautz. Vivid knowledge and tractable reasoning: Preliminary report. In *Proceedings of IJCAI-89*, pages 1146–1152, Detroit, MI, 1989.

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

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco, 1979.

[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.).

[Greiner and Schuurmans, 1992] Russ Greiner and Dale Schuurmans. Learning useful horn approximations. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR-92)*, pages 383–392, Cambridge, MA, 1992.

[Greiner, 1992] Russell Greiner. Learning near-optimal horn approximations. In *Preprints of the AAAI Spring Symposium on Knowledge Assimilation*. Stanford University, Stanford, CA, March 1992.

[Gupta and Nau, 1991] Naresh Gupta and Dana S. Nau. Complexity results for blocks-world planning. In *Proceedings of AAAI-91*, pages 629–635, Anaheim, CA, 1991.

[Heintze and Jaffar, 1990] Nevin Heintze and Joxan Jaffar. A finite presentation theorem for approximating logic programs. In *Proceedings of POPL-90*, pages 197–201, 1990.

[Henschen and Wos, 1974] L. Henschen and L. Wos. Unit refutations and horn sets. *Journal of the ACM*, 21(4):590–605, 1974.

[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*, pages 1121–1126, Detroit, MI, May 1989.

[Imielinski, 1987] Thomsz Imielinski. Domain abstraction and limited reasoning. In *Proceedings of IJCAI-87*, volume 2, pages 997–1002, 1987.

[Johnson, 1990] D. S. Johnson. A catalog of complexity classes. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 2. Elsevier Science Publishers B. V., North Holland, 1990.

[Johnson, 1993] D. S. Johnson, 1993. personal communication.

[Karp and Lipton, 1982] R. M. Karp and R. Lipton. Turing machines that take advice. *Enseign. Math.*, 28:191–209, 1982.

[Kautz and Selman, 1991] Henry Kautz and Bart Selman. A general framework for knowledge compilation. In *Proceedings of the International Workshop on Processing Declarative Knowledge (PDK)*, Kaiserslautern, Germany, July 1991.

[Kautz and Selman, 1992a] Henry Kautz and Bart Selman. Forming concepts for fast inference. In *Proceedings of AAAI-92*, pages 786–793, San Jose, CA, 1992.

[Kautz and Selman, 1992b] Henry Kautz and Bart Selman. Planning as satisfiability. In Bernd Neumann, editor, *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI-92)*, pages 359–364, Vienna, Austria, 1992.

[Kautz and Selman, forthcoming] Henry Kautz and Bart Selman. Efficient approximations of quantified formulas. (In Preparation), forthcoming.

[Kautz *et al.*, 1994] Henry A. Kautz, Michael J. Kearns, and Bart Selman. Horn approximations of empirical data. *Artificial Intelligence*, 74:129–145, 1994.

[Knoblock *et al.*, 1991a] Craig A. Knoblock, Steven Minton, and Oren Etzioni. Integrating abstraction and explanation-based learning in prodigy. In *Proceedings of AAAI-91*, pages 541–546, Anaheim, CA, 1991.

[Knoblock *et al.*, 1991b] Craig A. Knoblock, Josh D. Tenenberg, and Qiang Yang. Characterizing abstraction hierarchies for planning. In *Proceedings of AAAI-91*, pages 692–697, Anaheim, CA, 1991.

[Larrabee, 1992] T. Larrabee. Test pattern generation using boolean satisfiability. *IEEE Transactions on Computer-Aided Design*, January 1992.

[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.

[Levesque, 1984] Hector J. Levesque. A logic of implicit and explicit belief. In *Proceedings of AAAI-84*, pages 198–202, Austin, TX, 1984.

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

[Lewis, 1978] H.R. Lewis. Renaming a set of clauses as a horn set. *JACM*, 25(1):134–135, 1978.

[McAllester and Givan, 1992] David A. McAllester and Robert Givan. Natural language syntax and first-order inference. *Artificial Intelligence*, 56:1–20, 1992.

[Mitchell *et al.*, 1992] D. Mitchell, B. Selman, and H.J. Levesque. Hard and easy distribution of sat problems. In *Proceedings of AAAI-92*, San Jose, CA, 1992.

[Muggleton and Buntine, 1988] Stephen Muggleton and Wray Buntine. Machine invention of first-order predicates by inverting resolution. In J. Laird, editor, *Proceedings of the Fifth International Conference on Machine Learning*, pages 339–344, 1988.

[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*, 16:47–65, 1981.

[Reiter and de Kleer, 1987] Raymond Reiter and Johan de Kleer. Foundations of assumption based truth maintance systems: Preliminary report. In *Proceedings of AAAI-87*, pages 183–187, Seattle, WA, 1987.

[Reiter and Mackworth, 1989] R. Reiter and A. Mackworth. A logical framework for depiction and image interpretation. *Artificial Intelligence*, 41(2):125–155, 1989.

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

[Schoppers, 1987] M. J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *Proceedings of AAAI-87*, volume 2, pages 1039–1046, 1987.

[Selman and Kautz, 1991] Bart Selman and Henry Kautz. Knowledge compilation using horn approximations. In *Proceedings of AAAI-91*, pages 904–909, Anaheim, CA, 1991.

[Selman and Levesque, 1990] Bart Selman and Hector J. Levesque. Abductive and default reasoning: a computational core. In *Proceedings of AAAI-90*, pages 343–348, Boston, MA, 1990.

[Selman *et al.*, 1992] B. Selman, Levesque H.J., and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of AAAI-92*, pages 440–446, San Jose, CA, 1992.

[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, pages 416–422, Milan, Italy, 1987.

[Tseitin, 1968] G. S. Tseitin. On the complexity of derivation in propositional calculus. In A. O. Slisenko, editor, *Studies in Constructive Mathematics and Mathematical Logic, Part II*. 1968.

[Valiant and V.V., 1986] R.G. Valiant and Vazirani V.V. Np is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[Valiant, 1983] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1983.