This paper appears in
Proceedings of AAAI-94, Seattle, WA, July 1994.

An Empirical Evaluation of Knowledge Compilation

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

Abstract

Computational efficiency is a central concern in the design of knowledge representation systems. Compiling a knowledge-base into a more tractable form has been suggested as a way around the inherent intractability of many representation formalisms. Because not all theories can be put into an equivalent tractable form, Selman and Kautz (1991) have suggested compiling a theory into upper and lower bounds (one logically weaker, the other logical stronger) that approximate the original information.

A central question in this approach is how well the bounds capture the original knowledge. This question is inherently empirical. We present a detailed empirical evaluation of the compilation of two kinds of theories: computationally challenging randomly generated theories, and propositional encodings of planning problems. Our results show that one can answer a very high percentage of queries even using unit clause bounds, which are much easier to compute than more general tractable approximations. Furthermore, we demonstrate that many of the queries that can be answered by the bounds are expensive to answer using only the original theory: in other words, knowledge compilation does not just "skim off" easy queries. In fact, we show substantial total computational savings in using the bounds together with the original theory to answer all queries (with no errors) from a large benchmark set, over using the original theory alone. This study suggests that knowledge compilation may indeed be a practical approach for dealing with intractability in knowledge representation systems.

1 Introduction

In the design of knowledge representation systems, the tradeoff between expressive power and computational tractability has been studied extensively. Unfortunately, the languages that allow for efficient inference are often considered too restrictive. One way around this issue is to employ some form of *knowledge compilation*. The idea is to let the user enter statements into the knowledge base (KB) in an unrestricted language, and have the system subsequently translate the information into a tractable form. Since an exact translation is often not possible, Selman and Kautz (1991) propose to *approximate* the original theory by using two bounds, one logically weaker (the upper bound) and the other logically stronger (the lower bound). As an example, they consider compiling general propositional theories into two approximating Horn theories. Certain queries can be answered quickly by using the bounds, as will be described below.

Though theoretically appealing, the practical value of knowledge compilation will depend on how well the bounds approximate the original information. In other words, what fraction of the incoming queries can be answered quickly by using the bounds? We would also like it to be the case that among those queries that can be answered with the bounds, there are queries that *cannot* be answered easily using the original theory (*i.e.*, the bounds are not just "skimming off" the easy queries).

We will first show, by using a general complexity-theoretic argument, that there do exist theories for which answering certain queries using the bounds is much easier than answering the same queries on the original theory. This argument reveals the existence of such theories and queries, but does not rule out the possibility that one would rarely encounter them in practice. We therefore also undertook an empirical evaluation of the knowledge compilation approach. We considered two classes of theories: hard random theories and propositional encodings of planning problems. We compile those theories, and give experimental data which shows that the compilation leads to dramatic computational savings.

In order to conduct our experiments, we needed theories that were sufficiently challenging, so that answering queries would take a reasonable computational effort; otherwise there would be no need for compilation in the first place. For the random theories, we used the hard problem class as identified in Mitchell *et al.* (1992). For our planning problems, we constructed a simple autonomous robot domain. To our surprise, planning problems that would intuitively appear quite hard were often answered almost instantaneously by the standard Davis-Putnam satisfiability procedure (Davis & Putnam 1960). In fact, we were able to prove that a very general class of such problems can be solved in linear time by unit propagation (a standard component of satisfiable procedures), even though many AI planning systems would find them very difficult. After identifying this class of "easy" planning problems, we were able to construct a planning domain that is provably computationally difficult, as was needed in our evaluation of knowledge compilation.

The form of knowledge compilation examined in this paper is based on approximations between logical languages that fall into different classes in the hierarchy of computational complexity. The term "knowledge compilation" is used more broadly in the expert systems community to refer to a wide variety of work that aims to increase the efficiency of such systems. Much of this work develops techniques for transforming "deep" functional models of devices to "shallow" diagnostic rules (Chandrasekaran & Mittal 1983; Keller 1991). The output of such systems does not correspond to either an upper-bound or a lower-bound in our sense; while some information may be lost in the compilation process (as with our upper-bounds), the compilers themselves *introduce* domain-specific information about diagnosis. Others view knowledge compilation as a kind of automatic programming, with the goal of converting a system specification to an implementation that exactly satifies it (Dietterich 1991). Bylander (1991) provides a high-level logical characterization of some different kinds of knowledge compilation; interestingly, he argues that forms of knowledge compilation based on approximations (as is ours) are unlikely to provide significant computational improvement. However, his argument is based on the assumption that the compilation process itself must be tractable, which we explicitly reject.

2 Knowledge Compilation by Theory Approximation

Selman and Kautz (1991) define knowledge compilation by theory approximation as follows. Assume that we have a logical theory Σ. One can approximate Σ by two theories Σ_{lub} and Σ_{glb} that are in a given tractable logical language. The approximation is such that $\Sigma_{\text{glb}} \models \Sigma \models \Sigma_{\text{lub}}$. So, Σ_{glb} is logically stronger than the original theory, and is called a *greatest lower bound* (GLB); and Σ_{lub} is logically weaker than the original theory, and is called a *least upper bound* (LUB).[1] The bounds are the best ones possible, given the particular tractable language. This means, for example, that there does not exist a tractable theory Σ' that is not logically equivalent to the Σ_{glb} and is such that $\Sigma_{\text{glb}} \models \Sigma' \models \Sigma$. The LUB of a theory is unique, but there can be several distinct GLBs.

Let us consider an example of approximating a general propositional theory by two bounding Horn theories. We take $\Sigma = (\neg a \vee c) \wedge (\neg b \vee c) \wedge (a \vee b)$. ($a$, b, and c are propositional letters.) 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. These bounds can be verified by noting that

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

[1] The terminology is based on a model-theoretic view of the approximations. Note that the models of, for example, the Σ_{glb} form a subset of the models of Σ. We are interested in a largest possible subset. For another approach to approximating logical theories, see Dalal and Etherington (1992).

KC_Query(α)
if $\Sigma_{\text{lub}} \models \alpha$ then return "yes"
else if $\Sigma_{\text{glb}} \not\models \alpha$ then return "no"
else determine whether $\Sigma \models \alpha$ using
a general theorem prover and the original theory.

Figure 1: Fast querying using theory approximation. The original theory is Σ; Σ_{lub} and Σ_{glb} are its approximations; and α is the query.

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.

Instead of compiling into Horn theories, one can choose to compile into other tractable propositional theories, such as a set of unit clauses (*i.e.*, a conjunction of literals) or a set of binary clauses. Our experiments below show that even unit bounds lead to substantial computational savings.

Fig. 1 shows how the bounds can be used to improve the efficiency of a knowledge representation system. The system first tries to obtain an answer quickly by using the bounds, which can be done in linear time for Horn (Dowling & Gallier 1984) or unit bounds. In case no answer is obtained, the query is tested directly against the original theory. Note that KC_Query thus remains a *complete* procedure. A time-saving alternative would be for the system to simply return "don't know" if the bounds do not answer it.

The system can thus 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 approximations depends on how well the bounds characterize the original theory.

3 Computational Savings

The key question concerning knowledge compilation is whether it will lead to an actual savings in computational effort. For example, it could be the case that queries answered by the approximating bounds can also be answered quickly using the original theory. An obvious counterexample is any inconsistent theory. Compilation yields an inconsistent upper bound. Any query against this bound would quickly return "yes" (see Fig. 1). 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).

This complexity theoretic argument shows that there exist theories where compilation gives a provable computational savings. Of course, this still leaves open the question whether one would encounter such theories in practice. In the next two sections, we therefore present an empirical evaluation of knowledge compilation of two classes of theories. In both cases, we will demonstrate substantial computational savings.

4 Empirical Evaluation I: Hard Random Theories

In this section, we consider the compilation of hard, randomly-generated propositional theories. Mitchell *et al.* (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 & Putnam 1960).[2] However, Mitchell *et al.* also described how to 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 GLB of each. Table 1 gives the median size, in literals, of the LUB and GLB for each size theory. The bounds were computed using the algorithms as given in Selman and Kautz (1991), adapted for generating unit bounds. We generated the optimal

[2] 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: Statistics for compiling and querying hard random 3CNF theories.

bounds. Computation time for the unit LUBs ranged from 5 minutes for the 75 variable theories, to one hour for the 200 variable theories. (All experiments were run 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, as given in Table 1, 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 simplest 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

$$Pr((\Sigma_{\text{lub}} \vdash x \vee y) \text{ or } (\Sigma_{\text{glb}} \not\vdash x \vee y))$$

which equals

$$Pr(\Sigma_{\text{lub}} \vdash x \vee y) + Pr(\Sigma_{\text{glb}} \not\vdash x \vee y)$$

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,

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

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,

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

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 a similar calculation, we can calculate $Pr(\Sigma_{\text{lub}} \vdash x \vee y)$. 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.

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 can see from Table 1, the percentage of queries that can be handled by the unit clause bounds is quite high. Note that the queries handled by the bounds can be answered in linear time. The Davis-Putnam procedure, however, 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 hard 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. 1, 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".[3] 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 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.

[3]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 & Büning 1992; Dubois *et al.* 1993). Tableau (Crawford & Auton 1993) is one of the fastest implementations of the algorithm.

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 would increase even further when we would consider, for example, 10000 queries.) Thus in this case we have gone beyond the main objective of knowledge compilation, namely to speed query answering by shifting computational effort from on-line to off-line (compilation), and have actually reduced the total amount of work required.

Finally, we observe that 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.

5 Empirical Evaluation II: Planning Formulas

Planning has traditionally been formalized as first-order deduction (Green 1969; McCarthy & Hayes 1969). In this approach, a plan is basically a *proof* that a statement asserting the existence of a goal state is valid. Kautz and Selman (1992) develop an alternative formalization of planning as propositional satisfiability. They show how planning problems in typical domains, such as the blocks world, can be axiomatized so that every *model* of the axioms corresponds to a plan. The satisfiability formalization makes it easy to state facts about any state of the world (not just the initial and goal states) and is closer in spirit to modern constraint-based planners (Stefik 1981; Chapman 1987) than is the deductive approach.

We decided to evaluate knowledge compilation within the planning as satisfiability framework. The particular problems described in the Kautz and Selman paper all have unique models, corresponding to a single solution. Compiling such formulas provides no benefit beyond finding the single satisfying model. Therefore we developed a class of planning problems that each have many different solutions. Compiling these problems allows one to evaluate quickly various queries about what must hold in all solutions, as well as to pose queries that impose additional constraints on the possible solutions.

We call this domain the "mapworld". In the basic version of the mapworld, we imagine that a robot is moving between nodes of a graph, such as the one shown in Fig. 2. (Ignore for now the section of the figure labeled "MAZE", which will be explained later.) At each time step the robot can either stay in place or move to an adjacent node. An instance of the mapworld consists of axioms that describe a particular such graph, as well as constraints on the location of the robot at various times, up to some final instance; for example, that the robot be at node a at time 0 and at node g at final time 10. One can then pose queries to answer against these axioms, such as "Can the robot be at node f at time 2?" (obviously, no), or "Does the fact that the robot goes through node c imply that it does *not* go through node

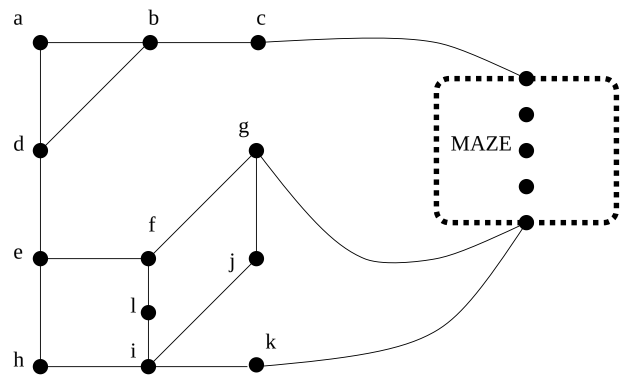

Figure 2: The mapworld domain.

k?" (less obviously, this implication does indeed hold, because it takes at least 11 steps to reach g when going through both c and k).

One application in which the ability to answer queries of this sort is useful is plan recognition (Schmidt, Sridharan, & Goodson 1978; Allen & Perrault 1980; Kautz 1986). For example, one may have partial knowledge about the goals and actions of another agent, and want to be able to infer the possible states the agent could be in at various times. Another interesting application is in reactive planning systems (Agre & Chapman 1987; Schoppers 1987; Kaelbling 1988; Kabanza 1990). An important issue in such systems is how to combine reactive behaviors (*e.g.* move to a node if it contains food) with more global plans (*e.g.* visit nodes x and y before the end of the day). A possible architecture for a combined system would have a reactive module that *proposed* actions (*e.g.* move to node c), which are however *rejected* if the axiomatization of the global planning problem entails the *negation* of the action (*i.e.*, the action would be incompatible with the global goals). This proposal is similar to Bratman, Israel, and Pollack's (1988) view of plans as *filters*, but note that we suggest filtering against the whole set of possible solutions (models) of the planning problem, rather than against a single one. Clearly it requires the ability to check rapidly if the proposed action is consistent with at least one solution to the global problem.

The basic mapworld can be captured by the following kinds of axioms. The proposition that the robot is a node x at time i is written x_i. First, there are movement axioms $\mathcal{M}$, that state that the robot always stays put or moves to an adjacent node; for our example, these would include

$$a_i \supset (a_{i+1} \vee b_{i+1} \vee d_{i+1})$$

for $0 \leq i < 10$. Second, there are disjointedness axioms $\mathcal{D}$, that state that the robot is only at one node at a time; for example,

$$a_i \supset (\neg b_i \wedge \cdots \wedge \neg k_i).$$

Finally, there are assertions that constrain the robot to be at certain locations at certain times; such positive assertions (such as $\{a_0, g_{10}\}$) are designated $\mathcal{P}$, while negative assertions (such as $\{\neg f_5\}$) are designated $\mathcal{N}$. Finally, we define propositions of the form $x_{\mathtt{ever}}$ by

$$x_{\mathtt{ever}} \equiv (x_0 \vee x_1 \vee \cdots \vee x_{10}).$$

Given these axioms, the first query above becomes "Does $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \mathcal{P} \models \neg f_2$?", and the second becomes "Does $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \mathcal{P} \models c_{\mathtt{ever}} \supset \neg k_{\mathtt{ever}}$?" Surprisingly, it turns out that *all* CNF queries against a basic mapworld problem can be answered quickly, using a standard theorem-prover.[4] One can prove that a simple rule of inference called "unit propagation" is complete for such theories. Unit propagation is takes only linear time, and is part of all resolution-style theorem proving procedures, such as the Davis-Putnam procedure. In general unit propagation by itself is not a complete decision procedure. However, one can prove the following theorem (Kautz & Selman 1994):

Theorem: *For any basic mapworld problem and clausal query α, we have*

$$\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \mathcal{P} \models \alpha$$

iff unit propagation proves that $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \mathcal{P} \cup \{\neg\alpha\}$ is inconsistent.

Thus, knowledge compilation is not needed in this case: unit propagation yields a linear-time decision procedure. This indicates that there are interesting computational advantages to using a satisfiability encoding for planning. For example, a standard STRIPS-style planner (Fikes & Nilsson 1971) would end up exploring (in general) an exponential number of paths before realizing that certain sets of nodes cannot be reached within a fixed time-bound.

To make our mapworld more computationally challenging, we generalize it by adding constraints that say that certain pairs of nodes are forbidden to appear on the same path. Such constraints often occur in real-life planning problems, where for example going through a node represents consuming some limited resource. An example of such a constraint is $\neg f_{\mathtt{ever}} \vee \neg j_{\mathtt{ever}}$, which states that the robot cannot pass through both nodes f and j on its way to g. This change makes planning much harder — in fact, answering CNF-queries becomes NP-complete, as can be shown by a reduction from *path with forbidden pairs* (Garey & Johnson 1979, page 203). This also greatly increases the applicability of our results, because most interesting planning problems are NP-complete (Gupta & Nau 1991; Erol, Nau, & Subrahmanian 1992) and thus can be efficiently encoded as mapworld problems.

In Fig. 2, the area labeled "MAZE" is a 30-node subgraph constructed so that all paths through it are blocked by various forbidden pairs of nodes. Disregarding

[4]It is not surprising, of course, that an efficient algorithm *exists* for these queries, because the lengths of the shortest paths between all points in a graph can be determined in polynomial time (Aho, Hopcroft, & Ullman 1974). What is unexpected is that the SAT encoding of the problem allows an efficient solution by a completely general theorem-prover, that does not employ a special shortest-path algorithm.

	RandBin	RandEver	Hand
number queries	500	400	5
theory only time	2013	8953	1071
KC_Query time	464	3748	439
Σ+LUB time	580	840	6.9
KC using Σ+LUB time	283	617	6.8
bounds only time	5	6	1
num. answered by bounds	376	144	2

Table 3: Statistics on querying mapworld with and without knowledge compilation. Time in seconds.

these pairs, the shortest path through the maze is 5 steps long. By counting alone, then, one would think that was possible to go from a to g by traversing the maze in no more than 10 steps (the time limit on the problem). It is computationally hard, however, to determine that these paths are blocked, and that in fact the robot *must* traverse the edge from d to e. Any query that depends on realizing this fact is also quite hard to answer. We therefore *compiled* the problem instance, effectively moving the most computationally difficult part of the reasoning off-line. (As we will see, the bounds also contain a great many other non-trivial conclusions concerning the mapworld example.)

The SAT encoding of the mapworld in Fig. 2 contains 576 variables and 29,576 clauses. It takes about 4400 seconds to compute both the unit LUB and a unit GLB. The unit LUB determines the values of 341 of the variables. The GLB we found was a single model, which nonetheless was useful in query-answering.

We then created three different test sets of queries: RandBin is a set of 500 random binary queries; RandEver is set of 400 random binary queries, where the propositions are taken just from the "ever" predicates; and Hand is a small set of hand-constructed queries that are intuitively interesting and non-obvious, such as $f_{\mathtt{ever}} \vee i_{\mathtt{ever}}$.

Table 3 compares the results of various ways of running the queries. For "theory only" the queries were simply tested against the uncompiled problem using tableau. The "KC_Query" row is the time required to answer all queries using the query algorithm presented in Fig. 1 that uses both the bounds and the original theory. In all cases we see a significant speed-up. In fact, the savings for the RandEver test set more than pays off the entire cost of computing the bounds.

We then experimented with several variations on the basic knowledge-compilation querying algorithm. For the "Σ+LUB" row we conjoined the original theory with its unit LUB, and then ran all queries using tableau. Note that the conjoined theory is logically unchanged (since the original theory entails its LUB), but is easier to reason with. For the RandBin test set, this approach is not as good as the plain "KC_Query" algorithm; however, for RandEver and Hand it is considerably faster.

Next, in the "KC using Σ+LUB" experiments we first tested each query directly against the bounds, but if they did not answer it, we then answered it using tableau with the conjoined theory. In every case this was the fastest complete method.

Finally, we ran the queries against the bounds only, leaving some of them unanswered. In all cases this took only a few seconds for hundreds of queries. About 75% of the RandBin queries, 36% of the RandEver queries, and 2 out 5 of the Hand queries can be answered in this way. However, the great difference in speed (*e.g.*, 5 seconds on the RandBin queries, versus 283 seconds for the fastest complete method) suggests that using the bounds alone may be most practical for many real-time applications. For example, in many domains instead of relying on expensive theorem proving a system may try to obtain information by direct sensing of its environment.

6 Conclusion

We have evaluated the computational savings that can be gained by compiling general logical theories into a pair of tractable approximations. We first argued on complexity-theoretic grounds that on certain theories knowledge compilation must result in computational savings. We then considered the compilation of two kinds of theories: hard random CNF theories, and propositional encodings of planning problems. In both cases our experiments showed that a high percentage of queries can be answered using the tractable bounds, and that the approach leads to a dramatic decrease in the time required to answer a large series of queries. This indicates that the knowledge compilation approach is useful for both unstructured, randomly generated theories, and highly structured theories such as encodings of planning domains. In this paper, we obtained good performance with unit clausal approximations. An open question that we will address in future work is whether it is worthwhile to compute the more accurate, but more expensive to obtain, Horn approximations. In any case, this study has shown that knowledge compilation by theory approximation is indeed a promising approach for dealing with intractability in knowledge representation systems.

References

Agre, P. E., and Chapman, D. 1987. Pengi: an implementation of a theory of activity. In *Proceedings of AAAI-87*, 268.

Aho, A. V.; Hopcroft, J. E.; and Ullman, J. D. 1974. *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley.

Allen, J. F., and Perrault, C. R. 1980. Analyzing intention in utterances. *Artificial Intelligence* 143–177.

Bratman, M. E.; Israel, D. J.; and Pollack, M. E. 1988. Plans and resource-bounded practical reasoning. *Computational Intelligence* 4(4). also SRI TR 425R.

Buro, M., and Büning, H. K. 1992. Report on a SAT competition. Technical Memorandum 110, Mathematik/Informatik Universität Paderborn.

Bylander, T. 1991. A simple model of knowledge compilation. *IEEE Expert* 6(2):73–74.

Chandrasekaran, B., and Mittal, S. 1983. Deep versus compiled knowledge approaches to diagnostic problem solving. *International Journal of Man-Machine Studies* 19(5):425–436.

Chapman, D. 1987. Planning for conjunctive goals. *Artificial Intelligence* 32:333–378.

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

Dalal, M., and Etherington, D. W. 1992. Tractable approximate deduction using limited vocabularies. In *Proceedings of CSCSI-92*, 206–212.

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

Dietterich, T. G. 1991. Bridging the gap between specification and implementation. *IEEE Expert* 6(2):80–82.

Dowling, W. F., and Gallier, J. H. 1984. Linear time algorithms for testing the satisfiability of propositional Horn formula. *Journal of Logic Programming* 3:267–284.

Dubois, O.; Andre, P.; Boufkhad, Y.; and Carlier, J. 1993. SAT versus UNSAT. In *Preprints, Second DIMACS Algorithm Implementation Challenge*. Piscataway, NJ: Rutgers University.

Erol, K.; Nau, D.; and Subrahmanian, V. 1992. On the complexity of domain-independent planning. In *Proceedings of AAAI-92*, 381–386.

Fikes, R. E., and Nilsson, N. J. 1971. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2.

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

Green, C. 1969. Application of theorem proving to problem solving. In *Proceedings of IJCAI-69*, 219–239.

Gupta, N., and Nau, D. S. 1991. Complexity results for blocks-world planning. In *Proceedings of AAAI-91*, 629.

Kabanza, F. 1990. Synthesis of reactive plans for multi-path environments. In *Proceedings of AAAI-90*.

Kaelbling, L. 1988. Goals as parallel program specifications. In *Proceedings of AAAI-88*, 60–65.

Kautz, H., and Selman, B. 1992. Planning as satisfiability. In *Proceedings of ECAI-92*, 359.

Kautz, H., and Selman, B. 1994. An empirical evaluation of knowledge compilation by theory approximation (extended version). Technical report, AT&T Bell Laboratories, Murray Hill, NJ.

Kautz, H. 1986. Generalized plan recognition. In *Proceedings of AAAI-86.*

Keller, R. M. 1991. Applying knowledge compilation techniques to model-based reasoning. *IEEE Expert* 6(2):82–87.

McCarthy, J., and Hayes, P. J. 1969. Some philosophical problems from the standpoint of artificial intelligence. In MICHIE, D., ed., *Machine Intelligence 4*. Chichester, England: Ellis Horwood. 463ff.

Mitchell, D.; Selman, B.; and Levesque, H. 1992. Hard and easy distribution of SAT problems. In *Proceedings of AAAI-92.*

Schmidt, C.; Sridharan, N.; and Goodson, J. 1978. The plan recognition problem: an intersectiuon of psychology and artificial intelligence. *Artificial Intelligence* 11.

Schoppers, M. J. 1987. Universal plans for reactive robots in unpredictable environments. In *Proceedings of AAAI-87*, volume 2, 1023.

Selman, B., and Kautz, H. 1991. Knowledge compilation using Horn approximations. In *Proceedings of AAAI-91*, 904–909.

Stefik, M. 1981. Planning with constraints (molgen: Part 1 and 2). *Artificial Intelligence* 16:111–170.

Valiant, R., and V.V., V. 1986. NP is as easy as detecting unique solutions. *Theoretical Computer Science* 47:85–93.

Appendix

Theorem: *For any basic mapworld problem and a clausal query α, we have*

$$\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \mathcal{P} \models \alpha$$

iff unit propagation proves that $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \mathcal{P} \cup \neg\alpha$ is inconsistent.

Proof: The "if" follows from soundness of unit propagation. To prove "only if", we first note that the negation of a clausal query is a set of unit clauses. Thus without loss of generality we can assume that these unit clauses have been added to $\mathcal{N}$ and $\mathcal{P}$, and that the query is simply the false clause, $\emptyset$. In other words, where $\vdash_{UP}$ is the unit propagation relation, we will show that $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \mathcal{P}$ is unsatisfiable only if $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \mathcal{P} \vdash_{UP} \emptyset$. We begin with the following lemma.

Lemma 1 *For any h_j, $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \models \neg h_j$ only if $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \vdash_{UP} \neg h_j$*

Proof by induction, where n is the highest time index.
Base case: $j = n$. Observe that $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \models \neg h_n$ iff $\neg h_n \in \mathcal{N}$, because otherwise the axioms are satisfied by the model that makes every variable false except for h_n.
Inductive case: assume the lemma is true j such that $i + 1 \leq j \leq n$. We will prove that it is true for $j = i$. So suppose $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \models \neg h_i$. Consider the following two cases:
(i) $\neg h_i \in \mathcal{N}$. Then plainly $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \vdash_{UP} \neg h_j$.
(ii) $\neg h_i \notin \mathcal{N}$. Let the move axiom for h_i be $\neg h_i \vee a_{i+1} \vee \cdots \vee z_{i+1}$. We claim that it must be the case that

$$\begin{gathered} \mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \models \neg a_{i+1} \\ \vdots \\ \mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \models \neg z_{i+1} \end{gathered}$$

This is so because we could otherwise construct a model of the axioms that satisfies h_i. For example, suppose there were a model M of the axioms that satisfied a_{i+1}. Then we can construct a model M' of the axioms satisfying h_i that corresponds to a path that starts at h_i and then goes through a_{i+1} and continues on as it does in M; that is, where we identify a model by the set of positive literals it satisfies,

$$M' = \{h_i\} \cup \{x_k | k \geq i + 1 \text{ and } x_k \in M\}$$

By the induction hypothesis, all of $a_{i+1}, \ldots, z_{i+1}$ follow from $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N}$ by unit propagation. Then unit propagation on these literals and the move axiom for h_i yields $\neg h_i$. This proves the lemma. ■

To prove the theorem we need only to generalize the lemma to allow the inclusion of a set of positive literals $\mathcal{P}$ among the axioms. Let us assume that $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \mathcal{P}$ is unsatisfiable.

First, suppose that there are a pair of literals $\{a_i, b_i\} \in \mathcal{P}$ that have the same index. Then $\emptyset$ easily follows by unit propagation from this pair and the disjointedness axioms $\mathcal{D}$. Therefore without loss of generality we can assume that there is no such pair. Furthermore, also without loss of generality we can assume that $\mathcal{P}$ is of of minimal size in order to make the axioms unsatisfiable. We will now show that this means that either $|\mathcal{P}| = 1$ or $|\mathcal{P}| = 2$.

Suppose (counter to fact) that $|\mathcal{P}| \geq 3$ and that for every subset $\mathcal{P}'$ of $\mathcal{P}$, that $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \mathcal{P}'$ is consistent. Let a_j be the second-highest indexed member of $\mathcal{P}$, and let

$$\begin{aligned}\mathcal{P}_1 &= \{x_i | x_i \in \mathcal{P}, i \leq j\} \\ \mathcal{P}_2 &= \{x_i | x_i \in \mathcal{P}, i \geq j\}\end{aligned}$$

Then there must be a model M_1 of $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \mathcal{P}_1$, and a model M_2 of $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \mathcal{P}_2$. Then we can create a model M of $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \mathcal{P}$ by simply concatenating the path described by M_1 up to a_j with the path described by M_2 starting from a_j:

$$M = \{x_k | x_i \in M_1, i \leq j\} \cup \{x_k | x_i \in M_2, i \geq j\}$$

Because this contradicts the assumption that the axioms are inconsistent, we can assume that there is no such $\mathcal{P}$ of minimal size larger than 2. We consider the two cases in turn:

(i) $|\mathcal{P}| = 1$. Let $\mathcal{P} = \{a_j\}$. It must be the case that $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \models \neg a_j$. Then, by the lemma, $\neg a_j$ follows from $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N}$ by unit propagation. So by one additional unit step of unit propagation, $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \mathcal{P} \vdash_{UP} \emptyset$.

(ii) $|\mathcal{P}| = 2$. Let $\mathcal{P} = \{a_j, b_k\}$ where $j < k$, where both $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \{a_j\}$ and $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \{b_k\}$ are satisfiable. Let $\mathcal{R}$ be the set of literals indexed by k that are "ruled out" by b_k and the disjointedness axioms $\mathcal{D}$: $\mathcal{R} = \{\neg x_k | x \neq b\}$ Note that $\mathcal{R}$ follows from $\mathcal{D} \cup \{b_k\}$ by unit propagation. Now, we claim that if $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \{a_j, b_k\}$ is unsatisfiable, then $\mathcal{M} \cup \mathcal{D} \cup (\mathcal{N} \cup \mathcal{R}) \cup \{a_j\}$ is also unsatisfiable. To show this, we will prove the contrapositive of the claim.

So suppose $\mathcal{M} \cup \mathcal{D} \cup (\mathcal{N} \cup \mathcal{R}) \cup \{a_j\}$ is satisfiable. Then there is a model M of this formula, describing a path that starts at a_j, and continues on through adjacent nodes all the way to time n. In particular, the path must go through some node at time k. Now, because this model satisfies $\mathcal{R}$, the only node this can be possibly be is b, so M satisfies b_k. Therefore M satisfies $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \{a_j, b_k\}$. This proves the claim.

Thus the claim lets us conclude that $\mathcal{M} \cup \mathcal{D} \cup (\mathcal{N} \cup \mathcal{R}) \cup \{a_j\}$ is unsatisfiable. But this formula fits the previous case (i), where the negative literals in $\mathcal{R}$ are simply added to $\mathcal{N}$. Therefore we know that $\mathcal{M} \cup \mathcal{D} \cup (\mathcal{N} \cup \mathcal{R}) \cup \{a_j\} \vdash_{UP} \emptyset$ which implies (since $\mathcal{R}$ follows by unit propagation from the original axioms) that $\mathcal{M} \cup \mathcal{D} \cup \mathcal{N} \cup \{a_j, b_k\} \vdash_{UP} \emptyset$. This completes the proof of the theorem. ∎