An Analysis of Approximate Knowledge Compilation

Alvaro del Val
Departamento de Ingenieria Informatica
Universidad Autonoma de Madrid
28049 Madrid, Spain
Email: delval@eii.uam.es

Abstract

Knowledge compilation is the process by which an initial theory Σ with respect to which inference is intractable is transformed into one or more "approximate" or equivalent theories with respect to which inference can be performed efficiently. Selman and Kautz introduced Horn lowest upper bound (LUB) approximations in [SK91], and generalized them in [KS91; SK95] to a number of target languages other than Horn. In this paper, we analyze the problem of knowledge compilation for arbitrary clausal target languages, generalizing in several ways previous results. We provide general characterizations of the LUB that are independent of the target language; analyze the properties of the Generate-LUB algorithm of Selman and Kautz, proving its correctness for any target language closed under subsumption (including a wide family of languages which guarantee *polynomial size* approximations); and generalize the procedure to arbitrary target languages. We also examine some computational aspects of these procedures and the quality of Horn approximations.

1 Introduction

Knowledge compilation is the process by which an initial theory £ with respect to which inference is intractable is transformed into one or more "approximate" or equivalent theories with respect to which inference can be performed efficiently. The notion of Horn approximations, more specifically of Horn lowest upper bound (LUB) and Horn greatest upper bound(s) (GLB), was introduced by Selman and Kautz in [SK91]. The idea is to map ("compile") a clausal propositional theory Σ into two Horn theories Σ_{glb} and Σ_{lub} such that Σ_{glb} is a weakest Horn theory that entails Σ, and Σ_{lub} is the strongest Horn theory entailed by Σ. These "approximate theories" can be used to efficiently answer queries about the consequences of Σ, as reviewed below. The key observation is that queries with respect to Σ_{glb} or Σ_{lub} can be answered in polynomial time, since both theories are Horn, rather than using an exponential algorithm to decide whether $\Sigma \models C$. Though the cost of obtaining Σ_{lub} and E_{glb} can and usually will be very high (which justifies thinking of it as a preprocessing or compilation step), this cost can be amortized over a sufficiently large set of queries about Σ_1 which can be answered more efficiently after compilation.

*Much of this work was performed while the author was a post-doc at Stanford University's Computer Science Department. Alternative email address: delval@C8.stanford.edu.

This framework was generalized by Kautz and Selman to approximations written in target languages other than Horn (see [KS91; SK95], and section 2 below). A key aspect of the framework is given by two procedures that generate the LUB and GLB in the given target languages, respectively the Generate-LUB and Generate-GLB algorithms. Selman and Kautz provide no proof of correctness for Generate-LUB either in its more restricted form (which generates Horn LUBs) or in the more general form in which it generates other kinds of LUBs. In this paper, we establish general conditions for Generate-LUB to be correct; these conditions are strictly weaker than those cited by Kautz and Selman, as the target language needs to be closed under subsumption, but not under resolution. We thus improve and correct Selman and Kautz's analysis of their general framework for knowledge compilation, greatly expanding the set of target languages for which the generic Generate-LUB algorithm yields correct results. This set now includes, in particular, any subset of K-CNF closed under subsumption, for example the k-Horn language [DP92]; an important feature of such languages is that the LUB has guaranteed polynomial size. In addition, a simple modification of the algorithm allows us to prove it correct with respect to *arbitrary* propositional clausal target languages.

The structure of this paper is as follows. In the next section, we review the main concepts of LUB and GLB knowledge compilation. Section 3 offers two general characterizations of the LUB for arbitrary target languages; in particular, completeness with respect to queries in the target language fully characterizes the LUB. Section 4 analyses the Generate-LUB algorithm, establishing general conditions for its correctness, and generalizing it to deal with arbitrary clausal target languages. In section 5 we consider some complexity issues, followed by an analysis of the quality of Horn approximations in section 6. Related work, together with some implications of our results on the goals of compilation, is discussed in the final section.

2 LUB approximations: Review

In this section, we review the framework for knowledge compilation, as described in [SK91; KS91]. The basic idea is to approximate a theory in a given source language by bounding from above and from below the set of models of the theory, where the bounds can be expressed in a computationally less difficult target language.

Kautz and Selman define a general framework for knowledge compilation in terms of arbitrary source and target languages and consequence relations. In this paper, we will focus only on clausal propositional and first order languages, with the classical consequence relation.

Let $\mathcal{L}$ be a propositional or first order clausal language. For any given "target language" $\mathcal{L}_T \subseteq \mathcal{L}$ we can define $\mathcal{L}_T$-upper and lower bounds of any theory Σ expressed in $\mathcal{L}$ as follows.

Definition 1 *For any $\Sigma \subseteq \mathcal{L}$, the theories $\Sigma_{lb}, \Sigma_{ub} \subseteq \mathcal{L}_T$ are respectively an $\mathcal{L}_T$-lower bound and an $\mathcal{L}_T$-upper bound of Σ iff $\Sigma_{lb} \models \Sigma \models \Sigma_{ub}$.*

Letting $Mod(\Gamma)$ denote the set of models of Γ, we have:

$$Mod(\Sigma_{lb}) \subseteq Mod(\Sigma) \subseteq Mod(\Sigma_{ub}),$$

Thus, Σ_{lb} approximates Σ "from below", whereas Σ_{ub} approximates Σ "from above". The "best" approximations are defined as follows:

Definition 2 *$\Sigma_{glb} \subseteq \mathcal{L}_T$ is an $\mathcal{L}_T$-*greatest lower bound *(GLB) of $\Sigma \subseteq \mathcal{L}$ iff $\Sigma_{glb} \models \Sigma$ and for no $\Sigma' \subseteq \mathcal{L}_T$ it is the case that $\Sigma_{glb} \models \Sigma' \models \Sigma$ but $\Sigma' \not\models \Sigma_{glb}$.*

A GLB of Σ is thus a weakest theory of the target language $\mathcal{L}_T$ that entails Σ. Similarly, a LUB of Σ is a strongest theory of $\mathcal{L}_T$ that is entailed by Σ:

Definition 3 *$\Sigma_{lub} \subseteq \mathcal{L}_T$ is an $\mathcal{L}_T$-*lowest upper bound *(LUB) of $\Sigma \subseteq \mathcal{L}$ iff $\Sigma \models \Sigma_{lub}$ and for no $\Sigma' \subseteq \mathcal{L}_T$ it is the case that $\Sigma \models \Sigma' \models \Sigma_{lub}$ but $\Sigma_{lub} \not\models \Sigma'$.*

We can use the $\mathcal{L}_T$-LUB Σ_{lub} and the $\mathcal{L}_T$-GLB Σ_{glb} of Σ for answering queries whether $\Sigma \models C$, for any clause C, as follows. If $\Sigma_{lub} \models C$ then $\Sigma \models C$, and if $\Sigma_{glb} \not\models C$ then $\Sigma \not\models C$ (otherwise, answer "don't know", or use a complete theorem prover to answer). Furthermore, it is a consequence of theorem 1 below that for $C \in \mathcal{L}_T$, if $\Sigma_{lub} \not\models C$ then $\Sigma \not\models C$. Thus, if the query language is a subset of $\mathcal{L}_T$ then only the $\mathcal{L}_T$-LUB is needed to answer queries. Of course, the use of the $\mathcal{L}_T$-LUB and $\mathcal{L}_T$-GLB instead of the original theory for answering queries only makes sense if one can reason more efficiently (either analytically or empirically) about the former than about the latter.

We will only be concerned with LUBs in this paper. Note that only the LUB can be used to derive logical consequences of Σ, as opposed to rejecting non-consequences. We speak of *the* $\mathcal{L}_T$-LUB, since it always exists (the empty theory is an $\mathcal{L}_T$-upper bound of any theory), and it must be unique up to logical equivalence: the conjunction (union) of any two upper bounds Σ' and Σ'' is by definition also an upper bound, which is at least as strong as both Σ' and Σ''. Hence the $\mathcal{L}_T$-LUB is equivalent to the conjunction of all $\mathcal{L}_T$-upper bounds. Further general characterizations of the $\mathcal{L}_T$-LUB are given below.

3 Characterizing the LUB

In this section we provide two alternative characterizations of the $\mathcal{L}_T$-LUB. We first prove a result that will be useful throughout the paper, namely: completeness for $\mathcal{L}_T$ queries about Σ is both sufficient and necessary for a theory of $\mathcal{L}_T$ to be the $\mathcal{L}_T$-LUB of Σ.

Theorem 1 *Let $\Sigma_{lub} \subseteq \mathcal{L}_T$, $\Sigma \subseteq \mathcal{L}$. The following statements are equivalent:*

- *Σ_{lub} is the $\mathcal{L}_T$-LUB of Σ.*
- *For every $C \in \mathcal{L}_T$: $\Sigma \models C$ iff $\Sigma_{lub} \models C$*

Proof ($\Rightarrow$) Suppose Σ_{lub} is the $\mathcal{L}_T$-LUB of Σ, and let $C \in \mathcal{L}_T$. If $\Sigma_{lub} \models C$ then $\Sigma \models \Sigma_{lub} \models C$. Suppose on the other hand that $\Sigma \models C$ yet $\Sigma_{lub} \not\models C$. Then $\Sigma' = (\Sigma_{lub} \cup \{C\})$ is such that $\Sigma' \subseteq \mathcal{L}_T$ and $\Sigma \models \Sigma' \models \Sigma_{lub}$, yet $\Sigma_{lub} \not\models \Sigma'$. This contradicts the fact that Σ_{lub} is the $\mathcal{L}_T$-LUB of Σ.

($\Leftarrow$) Suppose Σ_{lub} is *not* the $\mathcal{L}_T$-LUB of Σ. If $\Sigma \not\models \Sigma_{lub}$ then there exists $C \in \Sigma_{lub} \subseteq \mathcal{L}_T$ such that $\Sigma \not\models C$, where trivially $\Sigma_{lub} \models C$. Suppose therefore that $\Sigma \models \Sigma_{lub}$. Then Σ_{lub} is an $\mathcal{L}_T$-upper bound of Σ. If it's not the $\mathcal{L}_T$-LUB then there exists $\Sigma' \subseteq \mathcal{L}_T$ such that $\Sigma \models \Sigma' \models \Sigma_{lub}$ yet $\Sigma_{lub} \not\models \Sigma'$. Hence there exists $C \in \Sigma' \subseteq \mathcal{L}_T$ such that $\Sigma_{lub} \not\models C$ and $\Sigma \models \Sigma' \models C$. □

It is interesting to note that this characterization of the $\mathcal{L}_T$-LUB, which holds for arbitrary clausal target languages, opens up a somewhat different perspective on the goals of compilation. Namely, the choice of a target language can now be guided not (only) by efficiency in answering queries about the original theory, but by the desire to obtain completeness with respect to certain query languages, for some perhaps task specific purposes, while ignoring parts of the theory irrelevant to the task. We will briefly come back to this point at the end of the paper.

A second characterization of the $\mathcal{L}_T$-LUB can be given in terms of implicates of the source theory. For target languages closed under subsumption, the $\mathcal{L}_T$-LUB of Σ is logically equivalent to the set of prime implicates of Σ that are in $\mathcal{L}_T$.

Definition 4 *A clausal language $\mathcal{L}_T$ is* closed under subsumption *iff for every $C \in \mathcal{L}_T$, if a clause C' subsumes C then $C' \in \mathcal{L}_T$.*

Definition 5 *A clausal language $\mathcal{L}_T$ is* closed under resolution *iff for every $B, C \in \mathcal{L}_T$, if A is a resolvent of B and C then $A \in \mathcal{L}_T$.*

Theorem 2 *Let $\Pi(\Sigma)$ be the set of prime implicates of $\Sigma \subseteq \mathcal{L}$. If $\mathcal{L}_T$ is closed under subsumption then the $\mathcal{L}_T$-LUB of Σ is logically equivalent to $\Pi(\Sigma) \cap \mathcal{L}_T$.*

Proof This is a corollary of theorem 3 below. □

To see that theorem 2 does not hold in general for languages not closed under subsumption, let $\mathcal{L}_T$ be the language of definite clauses (clauses with exactly one positive literal). The $\mathcal{L}_T$-LUB of $\Sigma = \{\neg p, \neg q \vee r\}$ is equivalent to $\{\neg p \vee q, \neg q \vee r\}$, which is clearly *not* equivalent to $\Pi(\Sigma) \cap \mathcal{L}_T$.[1]

It is easy however to generalize theorem 2 to target languages not closed under subsumption. For this, we need the notion of prime $\mathcal{L}_T$-implicates.

Definition 6 *A clause C is an* $\mathcal{L}_T$-implicate *of Σ iff $C \in \mathcal{L}_T$ and $\Sigma \models C$. C is a* prime $\mathcal{L}_T$-implicate *of Σ iff C is an $\mathcal{L}_T$-implicate of Σ not strictly subsumed by any other $\mathcal{L}_T$-implicate of Σ. The set of prime $\mathcal{L}_T$-implicates of Σ is denoted $\Pi_{\mathcal{L}_T}(\Sigma)$.*

Theorem 3 *The $\mathcal{L}_T$-LUB of Σ is equivalent to $\Pi_{\mathcal{L}_T}(\Sigma)$.*

Proof By theorem 1, it suffices to show that for any $C \in \mathcal{L}_T$, $\Sigma \models C$ iff $\Pi_{\mathcal{L}_T}(\Sigma) \models C$. Let $C \in \mathcal{L}_T$. If $\Pi_{\mathcal{L}_T}(\Sigma) \models C$ then $\Sigma \models \Pi_{\mathcal{L}_T}(\Sigma) \models C$. For the other direction, if $\Sigma \models C$ then C is an $\mathcal{L}_T$-implicate of Σ, hence there exists $C' \in \Pi_{\mathcal{L}_T}(\Sigma)$ such that C' subsumes C. Thus $\Pi_{\mathcal{L}_T}(\Sigma) \models C' \models C$. □

Note that if $\mathcal{L}_T$ is closed under subsumption then $\Pi_{\mathcal{L}_T}(\Sigma) = (\Pi(\Sigma) \cap \mathcal{L}_T)$, from which theorem 2 follows as a corollary. More generally, $\Pi_{\mathcal{L}_T}(\Sigma)$ and therefore the $\mathcal{L}_T$-LUB of Σ can be characterized in terms of $\Pi(\Sigma)$ through a certain form of *weakening* of the prime implicates, a notion which can be seen as a dual of Selman and Kautz's notion of strengthenings.

Definition 7 *A non-tautologous clause C is an* $\mathcal{L}_T$-weakening *of a clause C' iff $C \in \mathcal{L}_T$, C' subsumes C, and no $C'' \in \mathcal{L}_T$ subsumed by C' strictly subsumes C. The set of $\mathcal{L}_T$-weakenings of a clause C is denoted $W_{\mathcal{L}_T}(C)$.*

For example, if $\mathcal{L}_T$ is the language of definite clauses over a vocabulary $\{p_1, \ldots, p_n\}$ then the set of $\mathcal{L}_T$-weakenings of $\neg p_1$ is $W_{\mathcal{L}_T}(\neg p_1) = \{\neg p_1 \vee p_i \mid 1 < i \leq n\}$. On the other hand, if $C \in \mathcal{L}_T$ then $W_{\mathcal{L}_T}(C) = \{C\}$. Finally, $\mathcal{L}_T$-weakenings may not always exist. If $\mathcal{L}_T$ is closed under subsumption then there are no $\mathcal{L}_T$-weakenings of any clause $C \notin \mathcal{L}_T$. There is for example no Horn-weakening of a non-Horn clause in the propositional case; and while Horn clauses have definite clause weakenings, non Horn clauses do not.

Despite these complexities, the next theorem shows that the $\mathcal{L}_T$-LUB can be obtained by collecting the $\mathcal{L}_T$-weakenings of the prime implicates of Σ ($\mu(\Gamma)$ denotes the result of removing subsumed clauses from Γ.)

Theorem 4 $\Pi_{\mathcal{L}_T}(\Sigma) = \mu(\bigcup_{C \in \Pi(\Sigma)} W_{\mathcal{L}_T}(C))$.

Proof ($\subseteq$) Suppose $C \in \Pi_{\mathcal{L}_T}(\Sigma)$. Then $\Sigma \models C$, hence there exists $C'' \in \Pi(\Sigma)$ that subsumes C. If $C'' \in \mathcal{L}_T$ then C subsumes C'' as well (otherwise $C \notin \Pi_{\mathcal{L}_T}(\Sigma)$, which would be a contradiction); hence $C \in \Pi(\Sigma)$, and since $C \in \mathcal{L}_T$, C is an $\mathcal{L}_T$-weakening of itself. Suppose on the other hand that $C'' \notin \mathcal{L}_T$. Because $C \in \Pi_{\mathcal{L}_T}(\Sigma) \subseteq \mathcal{L}_T$, if C is not an $\mathcal{L}_T$-weakening of C'' then there exists $C' \in \mathcal{L}_T$ subsumed by C'' which strictly subsumes C. Since $\Sigma \models C'' \models C'$, this contradicts $C \in \Pi_{\mathcal{L}_T}(\Sigma)$.

($\supseteq$) Suppose C is an $\mathcal{L}_T$-weakening of $C' \in \Pi(\Sigma)$, and C is not subsumed by any clause in the right hand side. Then $\Sigma \models C' \models C$ and $C \in \mathcal{L}_T$. Hence C is an $\mathcal{L}_T$-implicate of Σ, and clearly no other $\mathcal{L}_T$-implicate of Σ can strictly subsume C. □

We remark that all the results of this section hold for both propositional and first order clausal languages. In the rest of the paper we restrict our attention to propositional languages.

4 Computing the LUB

In this section, we provide procedures to compute the $\mathcal{L}_T$-LUB for *arbitrary* propositional clausal target languages. We begin by considering the case in which $\mathcal{L}_T$ is closed under subsumption, and later extend the results to the general case.

The following procedure computes an $\mathcal{L}_T$-LUB for any target language $\mathcal{L}_T$ that is closed under subsumption, as we will show; the procedure differs from the one given in [SK91] only in the incorporation of a tautology deletion strategy. The algorithm is a brute force resolution algorithm modified to avoid resolving together pairs of clauses both of which belong to the target language $\mathcal{L}_T$, in the expectation that this will lead to smaller compiled theories (smaller, that is, than what theorems2–4 would give us).

Procedure Generate-$\mathcal{L}_T$-LUB(Σ)

```
begin
$\Sigma_T := \{C \in \Sigma \mid C \in \mathcal{L}_T$ and $C$ is not tautologous$\}$
$\Sigma_N := \{C \in \Sigma \mid C \notin \mathcal{L}_T$ and $C$ is not tautologous$\}$
loop
  choose clauses $C_1 \in \Sigma_T \cup \Sigma_N$, $C_2 \in \Sigma_N$
      with a non-tautologous resolvent $C$ which
      is not subsumed by any clause in $\Sigma_T \cup \Sigma_N$
  if no such choice is possible then exit loop endif
  if $C \in \mathcal{L}_T$
  then delete from $\Sigma_T$ and $\Sigma_N$ any clause subsumed by $C$
         $\Sigma_T := \Sigma_T \cup \{C\}$
  else delete from $\Sigma_N$ any clause subsumed by $C$
         $\Sigma_N := \Sigma_N \cup \{C\}$
  endif
endloop
return $\Sigma_T$
end
```

In order to establish the correctness of the algorithm for target languages closed under subsumption we need two lemmas, whose proofs are omitted for lack of space.[2] In what follows, we write $\Gamma \vdash C$, for a clause C and a set of clauses Γ, iff there exists a resolution deduction from Γ of a clause C' that subsumes C; also, Σ_T and Σ_N refer to the final values of these variables in the Generate-$\mathcal{L}_T$-LUB algorithm.

Lemma 5 *If $\Sigma_T \vdash B$, $C \in \Sigma_N$, and A is a non-tautologous resolvent of B and C, then either $\Sigma_T \vdash A$ or there exists $A' \in \Sigma_N$ s.t. A' subsumes A.*

[1][SK91] prove theorems 1 and 2 for the Horn case. After writing this paper, we found a much more indirect proof of theorem 1 in [KR94].

[2]They are available from the author in a longer version of this paper.

This lemma in effect tells us that it is possible to transform certain resolution trees, so that all resolutions between two clauses of the target language occur in the bottom part of the transformed tree. Using these transformations, the next lemma in turn establishes that ET and EN completely characterize the set of clausal consequences of the source theory E.

Lemma 6 *If $\Sigma \vdash A$ then either $\Sigma_T \vdash A$ or there exists $A' \in \Sigma_N$ s.t. A' subsumes A.*

Suppose now that $\mathcal{L}_T$ is closed under subsumption. Then no $A' \in \Sigma_N$ can subsume A when $A \in \mathcal{L}_T$, so lemma 6 has the following immediate corollary:

Lemma 7 *Suppose $\mathcal{L}_T$ is closed under subsumption. If $\Sigma \vdash A$ and $A \in \mathcal{L}_T$ then $\Sigma_T \vdash A$.*

Theorem 8 *Generate-$\mathcal{L}_T$-LUB(Σ) computes the $\mathcal{L}_T$-LUB of Σ for any clausal language $\mathcal{L}_T$ closed under subsumption.*

Proof By theorem 1, it suffices to show that for any $C \in \mathcal{L}_T$, $\Sigma \models C$ iff $\Sigma_T \models C$. The right to left direction follows from the fact that $\Sigma \models \Sigma_T$. The other direction follows directly from lemma 7 and the completeness of resolution as an inference procedure. □

It was previously thought [SK95] that closure under resolution was also required for Generate-$\mathcal{L}_T$-LUB to be correct. That this is not the case is fortunate, since important languages such as k-Horn (Horn clauses with at most k literals) and k-quasi-Horn (clauses with at most k positive literals) are closed under subsumption but not under resolution. Important uses of k-Horn approximations are discussed in [DP92; KS92b]. More generally, any subset of k-CNF closed under subsumption (which includes k-Horn) can now be the target language for Generate-$\mathcal{L}_T$-LUB. *Note that the use of any such subset as target language guarantees that the compiled approximate theory has polynomial size.*

Let us now turn our attention to target languages which are *not* closed under subsumption. One could use theorem 3 to compute the $\mathcal{L}_T$-LUB for any such $\mathcal{L}_T$: compute all prime implicates, and then collect those in $\mathcal{L}_T$ plus the $\mathcal{L}_T$ weakenings of those not in $\mathcal{L}_T$. Fortunately, the space optimization used in the Generate-$\mathcal{L}_T$-LUB algorithm can still be used. The next theorem implies that, for *any* target language $\mathcal{L}_T$, we can compute the $\mathcal{L}_T$-LUB of an arbitrary clausal theory Σ by computing Σ_T and Σ_N using the original Generate-$\mathcal{L}_T$-LUB algorithm, returning the union of Σ_T with the set of $\mathcal{L}_T$-weakenings of clauses in Σ_N;[3] resolutions within $\mathcal{L}_T$ can still be avoided for such languages.

Theorem 9 *For any $\Sigma \subseteq \mathcal{L}$, the theory $\Sigma_{lub} = (\Sigma_T \cup \bigcup_{C \in \Sigma_N} W_{\mathcal{L}_T}(C))$ is the $\mathcal{L}_T$-LUB of Σ.*

Proof By theorem 1, it suffices to show that for any $C \in \mathcal{L}_T$, $\Sigma \models C$ iff $\Sigma_{lub} \models C$. The right to left direction follows from the fact that $\Sigma \models \Sigma_{lub}$. For the other direction, let $C \in \mathcal{L}_T$ and suppose $\Sigma \models C$. By lemma 6, either $\Sigma_T \models C$ (in which case trivially $\Sigma_{lub} \models C$), or there exists $C' \in \Sigma_N$ such that C' subsumes C. In the latter case, since $C \in \mathcal{L}_T$, there exists an $\mathcal{L}_T$-weakening C'' of C' such that C'' subsumes C (possibly $C'' = C$). Since $C'' \in W_{\mathcal{L}_T}(C')$ and $C' \in \Sigma_N$, $\Sigma_{lub} \models C'' \models C$. □

Note that theorem 8 is a special case of theorem 9, since if $\mathcal{L}_T$ is closed under subsumption then $\bigcup_{C \in \Sigma_N} W_{\mathcal{L}_T}(C) = \emptyset$ (see observations after definition 7). We thus conclude that there exists a procedure that computes the $\mathcal{L}_T$-LUB for arbitrary clausal target languages.

5 Computational complexity

The perhaps surprising power of the Generate-$\mathcal{L}_T$-LUB algorithm is not without serious costs, however. As shown in this section, the space requirements of the procedure are exponential in the combined size of input and output.

Before proving this result, it is worth mentioning first the following simple generalization of theorem 1 from [SK91], which has an essentially identical proof.

Theorem 10 *If $\mathcal{L}_T$ can express unsatisfiable theories[4] then the $\mathcal{L}_T$-LUB of Σ is satisfiable iff Σ is satisfiable.*

It does not follow from this, in general, that computing the $\mathcal{L}_T$-LUB is intractable (consider the degenerate case in which $\mathcal{L}_T = \mathcal{L}$). It does follow however that for any such $\mathcal{L}_T$ either there is no polynomial time algorithm for computing the $\mathcal{L}_T$-LUB, or inference in $\mathcal{L}_T$ cannot be performed in polynomial time (provided $P \neq NP$). For if both tasks could be performed in polynomial time we could decide satisfiability in polynomial time.

Let us now turn to the space requirements of Generate-$\mathcal{L}_T$-LUB, the main goal of this section. An unfortunate consequence of lemma 6 is that if $\mathcal{L}_T$ is closed under resolution then Σ_N contains all prime implicates of Σ not in $\mathcal{L}_T$.

Theorem 11 *If $\mathcal{L}_T$ is closed under resolution then $(\Pi(\Sigma) \setminus \mathcal{L}_T) \subseteq \Sigma_N$.*

Proof Suppose $C \in (\Pi(\Sigma) \setminus \mathcal{L}_T)$. Then $\Sigma \vdash C$, hence by lemma 6 either $\Sigma_T \vdash C$ or there exists $C' \in \Sigma_N$ such that C' subsumes C.

Case 1: $\Sigma_T \vdash C$. Since $\mathcal{L}_T$ is closed under resolution and $\Sigma_T \subseteq \mathcal{L}_T$, $\Sigma_T \vdash C'$ for some $C' \in \mathcal{L}_T$ that subsumes C. But since $C \notin \mathcal{L}_T$ by hypothesis, C' must *strictly* subsume C. Since $\Sigma \models \Sigma_T \models C'$, this contradicts $C \in \Pi(\Sigma)$. Hence this case is impossible.

Case 2: there exists $C' \in \Sigma_N$ that subsumes C. Since $\Sigma \models \Sigma_N \models C'$, and $C \in \Pi(\Sigma)$, C must subsume C' as well, hence $C \in \Sigma_N$. □

The converse inclusion does not hold. For a counterexample, let $\mathcal{L}_T$ be the language of Horn clauses, and let $\Sigma = \{p \vee \neg q, q \vee \neg r, p \vee \neg r \vee s\}$. Then $\Sigma_N = \{p \vee \neg r \vee s\} \not\subseteq$

[3] Except that in the "else" clause we should also delete from ET any clause subsumed by *C* if we do not want to end up with some subsumed clauses.

[4] The language of definite clauses, for example, cannot express unsatisfiable theories: the valuation which assigns *true* to every symbol satisfies any set of definite clauses. On the other hand, any language closed under subsumption includes the empty clause.

$\Pi(\Sigma)$. Nor does the theorem hold in general for languages which are not closed under resolution: let $\mathcal{L}_T$ be the language 3-Horn, with $\Sigma = \{\neg p \vee \neg q \vee \neg r, r \vee \neg s \vee \neg t\}$. Then $\Sigma_N = \emptyset$, yet $\Pi(\Sigma) \setminus \mathcal{L}_T = \{\neg p \vee \neg q \vee \neg s \vee \neg t\}$.

Consider now any theory with an exponential number of prime implicates (examples can be found in [KT90; CM78]). By adding two *new* positive literals to every clause, any such theory will have an exponential number of *non-Horn* prime implicates, since those two literals will occur in any clause entailed by the theory; in fact, all prime implicates will be non-Horn. And since the language of Horn clauses is closed under resolution, all these prime implicates will be computed and stored by Generate-L_T-LUB, for LT — Horn, despite the fact that the Horn-LUB of any such theory will be empty. Similar or identical examples can be used to show that the same holds for many other target languages. These include, to begin with, subsets of the Horn language, such as definite clauses, k-Horn clauses (Horn with at most *k* literals, for any fixed k), and unit clauses (at most one literal per-clause); but also languages such as reverse Horn (clauses with at most one negative literal) and its subsets, binary clauses (clauses with at most two literals), or a language that allows only a subset of the symbols of the language. The following corollary summarizes this discussion.

Corollary 12 *The Generate-LT-LUB algorithm requires exponential space (hence time), even in cases in which it outputs the empty theory as the CT-LUB.*

Cases in which Generate-L_T-LUB has exponential size output have already been described in [KS92a], Corollary 12 is stronger in that the exponential space requirements are not justified by the size of the output, in other words, time and space are exponential in the *combined size* of input and output. Notice furthermore that for all the above mentioned theories with empty LUB, Generate-LT-LUB reduces to a brute force prime implicate algorithm, which is likely to be extremely inefficient.

There are other complexity results from the literature on Horn approximations that readily generalize to a wide variety of target languages. In particular, the proof [KS92a] that Horn approximations most likely require worst case exponential space (unless non-uniform $P \subseteq NP$) applies with little or no modification to many other target languages, as was already noted in [KR94]. Another result that can be generalized deals with the complexity of inference with respect to the $\mathcal{L}_T$-LUB, that is of deciding whether $\Sigma_{lub} \models C$, given as input Σ and C. The reason behind this question is that we may want to consider this inference problem without explicitly computing the $\mathcal{L}_T$-LUB. Cadoli [Cad93] shows that this problem has exactly the same complexity as that of deciding whether $\Sigma \models C$, for Horn C. Using the notion of $\mathcal{L}_T$-strengthening introduced in [KS91] as a generalization of Horn strengthenings [SK91], one can easily generalize this result to arbitrary target languages closed under resolution

6 The quality of approximations

There are at least two important aspects in assessing the quality of an approximate theory: its size, and its "closeness" to the original theory. In this section, we discuss the latter from a worst case perspective. For concreteness, we will focus on Horn approximations. We show that the weakening of the original theory represented by the LUB can result in failing to answer an exponential number of queries, and in adding an exponential number of models to those of the original theory. We leave the reader the task of identifying further target languages for which the conclusions of this section apply.

Example 1 The examples used to establish corollary 12 have an exponential number of prime implicates which do not follow from the Horn LUB. There is therefore an exponential number of queries entailed by the original theory that the LUB will fail to answer.

Example 2 Given a set of variables $x_1, \ldots, x_n$, let $\hat{\Sigma} = \{x_i \vee x_j \mid 1 \leq i < j \leq n\}$ be the set of all binary positive clauses on this vocabulary. This theory is satisfied by exactly the set of interpretations that satisfies at most one negative literal, hence it has $n+1$ models. The Horn LUB is empty, however, so it has 2^n models. The ratio between the number of models of the LUB and the number of models of Σ is therefore exponential. A similar asymptotic ratio holds for any fixed k, with $\Sigma = \{k\text{-ary positive clauses}\}$.

In the last example, it is possible to work around the problem by observing that Σ can be brought to Horn form by an uniform renaming of all symbols, i.e. Σ is in the class "renamable Horn" and is therefore tractable. The next example does not have this property.

Example 3 Given variables $x_1, \ldots, x_n$, the non-parity theory is given by the sentence $\neg(x_1 \oplus \ldots \oplus x_n)$, where $\oplus$ denotes the exclusive-or connective. The models of this theory are all interpretations that satisfy an even number of *positive* literals, for a total of 2_{n-1} models. In prime implicate form, it can be written as

$$\bigwedge_{\{l_1,\ldots,l_n\} \in O^-} (l_1 \vee \ldots \vee l_n)$$

where *0~* is the set whose elements are those sets consisting of exactly one literal for each variable in the language, such that the total number of *negative* literals is odd. (See the discussion of the parity function in [Weg87].) If n is odd (the case with *n* even is left to the reader), then the Horn LUB approximation of non-parity, equivalently the set of Horn prime implicates of non-parity, contains the single clause $\neg x_1 \vee \ldots \vee \neg x_n$ (any other implicate contains at least two more positive literals). This clause rules out exactly one model, hence the LUB has 2^n - 1 models, or 2^{n-1} - 1 more models than the original theory. Any model containing at least one negative literal will satisfy the LUB, rather than only those models with an even number of positive literals.

Example 4 There is also at least one fairly natural class of theories that yield inadequate approximations.

Consider the propositional encoding of constraint satisfaction problems (CSP). For each variable X^j of a given CSP, its propositional encoding Σ contains a clause $C_j = (X_j = x_1^j \vee \ldots \vee X_j = x_{n_j}^j)$ specifying its initial domain, clauses specifying that distinct values of a variable are incompatible, and where n-ary constraints for $n \geq 2$ are encoded by negative clauses ("nogoods") stating which combinations of values are incompatible.

For each clause C_j, there must exist a unique clause $C_j' \in \Pi(\Sigma)$ which subsumes C_j. Now, it is easy to see that Σ_{lub}, being Horn, can only entail C_j' iff C_j' is a unit clause, i.e. if the CSP uniquely determines a single possible value for the given variable. For any variable for which this is not the case, the information that the variable must have at least one value *will be lost* by the Horn LUB, and the LUB will be consistent with a number of variables having no value. If k is the number of variables for which the CSP does not uniquely determine a value, this gives at least $\Sigma_{j=1}^{k} \binom{n}{k}$ additional models. Independently of quantitative measures, moreover, we would argue that there is a qualitative sense in which these approximations are inadequate — for example, if *Xj* denotes the position of an object, the compiled theory may be consistent with the object mysteriously vanishing (i.e. with *Xj* having no value), so a robot may have no reason to look for it. Note incidentally that combining the LUB with a GLB would be of little help for such problems, as any GLB of Σ must entail a complete assignment of values to every variable (because all Horn strengthenings of the various clauses *Xj* are positive unit clauses.)

In conclusion, it is easy to find examples which defeat Horn approximations, even in theories (such as CSPs) with a very small proportion of non-Horn clauses. While this is an important fact, we emphasize that it in no way precludes a profitable use of Horn approximations for large classes of theories which do not include the ones discussed here.

7 Discussion

Tn this paper, we have provided an analysis of approximate knowledge compilation on the basis of the approach introduced by Selman and Kautz in [SK91; KS91; SK95]. Highlights of the analysis, which generalize the results of Selman and Kautz in a number of ways, are:

- the characterization of the L_T-LUB for arbitrary clausal languages used as target of the compilation, both in terms of completeness with respect to LT-queries and of prime L_T-implicates;
- an analysis and proof of correctness of the generic procedure Generate-L_T-LUB, as introduced in [KS91], showing that it can be used without modification for any target language closed under subsumption;
- for languages not closed under subsumption, we have shown that the main computational attractive of the procedure, namely the avoidance of resolutions among clauses of the target language, can be preserved, providing a new generic algorithm that computes the LT-LUB for arbitrary propositional clausal languages.

As already mentioned, there is an important class of languages, closed under subsumption but not under resolution, for which, contrary to what was previously thought, GenerateLT-LUB gives correct results. This class of languages includes any subset of k-CNF closed under subsumption; the LUB with any such subset as target language has polynomial size. Of special interest among them is the language k-Horn, which is tractable, and which has been explored in detail as an approximation tool in [DP92; KS92b].

We have also analyzed some points which are crucial to the evaluation of the concepts and procedures discussed. First, either computation of the Lt-LUB, or inference in L*T* is likely to be intractable. Second, for many target languages Generate-L_T-LUB has exponential space and time requirements even in cases where the L_T-LUB is the empty theory. Finally, we have analyzed the quality of Horn approximations in terms of closeness to the initial theory.

In the category of related work, the great debt of this paper to the work of Selman and Kautz should be obvious to any reader; credit for specific results or proofs due to them has been explicitly indicated where appropriate. We should also mention that our results may have consequences for other approaches to knowledge compilation, specifically for the work of del Val in [dV94]. del Val presents procedures to compile propositional theories into *equivalent,* not just approximate, theories for which unit resolution is complete. One of these procedures uses the skeleton of the Generate-L_T-LUB algorithm as instantiated for the Horn target language, and the question arises whether the cited procedure can be generalized to take advantage of the results of this paper.

Finally, let us mention the work of Inoue [Ino92] on linear resolution procedures for finding the "characteristic clauses" (prime L_T-implicates) of a "production field" (target language L*T)*- Just like us, Inoue considers the problem for arbitrary target languages, providing procedures that compute the characteristic clauses of a theory for any target language closed under subsumption (what he calls an "stable" production field). Interestingly, he discusses multiple applications of the notion of characteristic clauses, which suggest a somewhat different perspective on the goals of compilation. For example, in abduction we are interested in certain kinds of entailments of the database, namely the implicates that involve only literals describing allowable hypothesis (abducibles) and some literal to be explained in terms of those hypothesis. Similarly, in determining the circumscriptive consequences of a propositional theory we are interested in particular in clauses that involve only positive literals whose symbol is being "minimized" or literals made from the set of "fixed", non-variable symbols. In diagnosis, we are interested in the "minimal conflicts", that is, the prime implicates that contain only AB-literals. In either case, the desired class of implicates can be designated as target language $L_{T;}$ the Generate-L_T-LUB procedure can then be used to ensure completeness for queries expressed in L_T- One can also think of less sophisticated but equally useful task specific reasons for choosing the target language. For example, we may want to compile the input-output behavior of a device assembled by means of some other more elementary components described propositionally. One possibility is to compute the prime implicates of the theory resulting from combining the theories corresponding to the device's components, throwing out all those which involve "internal" variables which do not refer to the initial input or final output of the assembled device. The other possibility, which may require much less space, is to designate the clauses involving only input and output variables as target language, and compile using the standard Generate-L_T-LUB.

In summary, achieving completeness with respect to a given target or query language is a worthwhile goal for LUB compilation. Even if the query language is not tractable, one can benefit from ignoring irrelevant parts of the initial theory, and there is always the possibility of further compiling the LUB into a tractable form in a second pass, possibly using some other method. There is in fact an interesting alternative when the query language $\mathcal{L}_Q$ is not tractable but its *complement* is closed under resolution. In this case, we can choose the complement as target language L_T for Generate-L_T-LUB, and obtain in Σ_N the prime implicates of Σ which belong to L_Q (as implied by theorem 11), hence the *CQ-LUB.* This guarantees completeness and tractability (relative to the size of Σ_N) with respect to L_Q, while avoiding all resolutions among clauses which do not belong to the query language. As an example, one can obtain the k-quasi-Horn LUB, in prime implicate form, by designating its complement as target language; no pairs of non k-quasi-Horn clauses will ever be resolved together by Generate-L_T-LUB in this case.

Both Inoue's linear resolution procedure and the procedures of this paper can be used to obtain completeness with respect to the selected query language (Inoue's results are limited in this regard to languages closed under subsumption, though the results of the present paper can be easily used to lift this restriction). There are two main differences. First, the restrictions imposed by his version of linear resolution have no analogous in Generate-L_T-LUB; the latter is therefore much more likely to generate redundant resolution derivations. However, and this is the second crucial difference, his procedure does compute *all* prime L_T-implicates, producing "compiled" theories which in general will require much more space than those generated by the LUB algorithm; this fact limits the usefulness of the linear resolution procedures for compilation purposes.

References

[Cad93] Marco Cadoli. Semantic and computational aspects of horn approximations. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence,* 1993.

[CM78] Ashok K. Chandra and George Markowsky. On the number of prime implicants. *Discrete Mathematics,* 24:7-11, 1978.

[dV94] Alvaro del Val. Tractable databases: How to make propositional unit resolution complete through compilation. In J. Doyle, E. Sandewall, and P. Torassi, editors, *KR'94, Proceedings of* ***Fourth International Conference on Principles of Knowledge Representation and Reasoning,*** **pages** 551-561, 1994.

[DP92] Rina Dechter and Judea Pearl. Structure identification **in relational data.** ***Artificial Intelligence,*** 58:237-290,1992.

[Ino92] Katsumi Inoue. Linear resolution for consequence-finding. ***Artificial Intelligence,*** **56:301-353, 1992.**

[KR94] Roni Khardon and Dan Roth. Reasoning with **models. In** ***AAAI'94, Proceedings of the Twelfth National American Conference on Artificial Intelligence,*** pages 1148-1153, 1994.

[KS9l] Henry Kautz and Bart Selman. A general framework for knowledge compilation. In *Proceedings* ***of the International Workshop on Processing Declarative Knowledge (PDK),*** **1991.**

[KS92a] Henry Kautz and Bart Selman. Forming concepts for fast inference. In *Proceedings of the* ***Tenth Conference of the AAAI,*** **1992.**

[KS92b] Henry Kautz and Bart Selman. Horn approximations **of empirical data.** ***Artificial Intelligence,*** 1992.

[KT90] Alex Kean and George Tsiknis. An incremental method for generating prime implicants/implicates. ***Journal of Symbolic Computation,*** 9:185-206, 1990.

[SK9l] Bart Selman and Henry Kautz. Knowledge compilation using Horn approximations. In *Proceedings* ***of the Ninth Conference of the AAAI,*** **1991.**

[SK95] Bart Selman and Henry Kautz. Knowledge compilation and theory approximation. *Journal of the ACM,* in press, 1995.

[Weg87] Ingo Wegener. ***The Complexity of Boolean*** *Functions.* Wiley, 1987.