Approximate Knowledge Compilation: The First Order Case

Alvaro del Val
Departamento de Ingeniería Informática
Universidad Autónoma de Madrid
28049 Madrid
delval@ai.ii.uam.es
http://www.ii.uam.es/~delval

Abstract

Knowledge compilation procedures make a knowledge base more explicit so as make inference with respect to the compiled knowledge base tractable or at least more efficient. Most work to date in this area has been restricted to the propositional case, despite the importance of first order theories for expressing knowledge concisely. Focusing on (LUB) *approximate* compilation (Selman and Kautz 1991), our contribution is twofold:

- We present a new ground algorithm for approximate compilation which can produce exponential savings with respect to the previously known algorithm (Selman and Kautz 1991).
- We show that both ground algorithms can be lifted to the first order case preserving their correctness for approximate compilation.

Introduction

Knowledge compilation procedures make a knowledge base (logical theory) Σ more explicit so as make inference with respect to the compiled knowledge base $\Sigma^\star$ tractable, or at least more efficient. The key idea is to invest time and space in an extra preprocessing effort which will later substantially speed up query answering, in the expectation that the cost of compilation will be amortized over many queries. Knowledge is acquired and stored so as to be reused, so improving the efficiency of each occasion of use of knowledge is evidently a worthy goal. This is specially so because of the large potential for redundant work during the lifecycle of query-intensive knowledge bases.

Compilation can be *exact*, or equivalence-preserving, when $\Sigma^\star$ is equivalent to the source Σ, thus guaranteeing efficient answers to all possible queries (see (del Val 1994, Marquis 1995)). Alternatively, compilation can be *approximate*: we only require that the compiled theory "approximates" the source in some precise sense, and as result only a subset of the possible queries can be efficiently processed. In this paper, we will be concerned with approximate compilation, specifically with the approach introduced in (Selman and Kautz 1991, Selman and Kautz 1995) of computing the Lowest Upper Bound (LUB) of a theory Σ as a way to "approximate" Σ in some fixed "target language" $\mathcal{L}_T$. Our contribution is twofold.

First, we develop a new algorithm (GLUB-2) for computing ground LUB approximations, and show it correct for target languages closed under subsumption whose complement is closed under resolution. As shown in (del Val 1995), only closure under subsumption is needed for the correctness of the original algorithm (GLUB-1) (Selman and Kautz 1991), so the new algorithm has a more restricted scope of applicability; nevertheless, some of the most important target languages for approximate compilation, such as Horn, k-quasi-Horn (clauses with at most k positive literals, for a fixed k), and others, satisfy this additional restriction, as discussed below. The new GLUB-2 can be exponentially more efficient than the old GLUB-1.

Second, we show that the main results on the properties of both algorithms for computation of the $\mathcal{L}_T$-LUB can be lifted to the predicate calculus under quite general conditions, with only some relatively mild additional assumptions about the behavior under instantiation resulting from the choice of target language.

Almost all work in knowledge compilation to date considers only the propositional case. One reason for this is that first order compilation raises a number of significant and subtle issues such as termination of the compilation algorithms, finiteness of the compiled theory, and semidecidability of inference, which do not appear in propositional compilation and which are not as important for standard first order automated reasoning as they are for compilation. Nevertheless, the need for a first order analysis of knowledge compilation is evident from the richer expressivity of predicate calculus. Even more than expressivity, we are often after *conciseness*. As an example common in the database and AI literature and practice, consider the case in which we assume a finite domain of objects. This can be expressed by a domain closure axiom, or less restrictively through an axiom expressing a fixed

finite upper limit on the cardinality of the universe of any satisfying interpretation. Any finite FOL theory Σ supplemented with such an axiom can be replaced by an essentially equivalent finite "propositional" theory consisting of ground clauses, possibly after the addition of new constants. But the size of this propositional theory can easily grow exponentially with respect to the initial FOL theory, as any n-ary predicate can yield up to D^n ground atomic instances for a domain of size D. The problem is even more acute for model-based representations (Kautz and Selman 1992, Khardon and Roth 1994), as the size of the description of just a single model can also grow exponentially, and the number of models doubly-exponentially. So even with finite domains the use of purely propositional compilation on propositional theories is computationally unfeasible, specially in comparison with the cost of compiling equivalent first order theories.

In the category of related work, (Selman and Kautz 1995) successfully deal with first order GLB (greatest lower bound) compilation. Also close to the approach studied in this paper, for example in its emphasis on "consequence finding" and restricted target languages, is the paper by (Inoue 1992). Some discussion of this work in connection with LUB approximations can be found in (del Val 1995). Inoue's method attempts to *actually compute* all the (target language) consequences of the input theory, even though there might be an infinite number of them. In contrast, the purpose of LUB approximations and other forms of compilation is only to ensure that all the (target language) consequences are *efficiently derivable* from the compiled theory.

The structure of this paper is as follows. After some initial definitions which are useful for characterizing languages, we review the notion of LUB approximation. Then we present the new ground algorithm, and subsequently we show that the results on the properties of both ground LUB algorithms can be lifted to predicate calculus. We consider next the types of target languages for which the algorithms are suitable, and we end with some discussion of issues in first order compilation.

Definitions

For our purposes, a language is simply a set of clauses over some fixed vocabulary of predicate symbols and term constructors. Alphabetic variants of clauses are regarded as identical. We use $\mathcal{L}$ for the "complete language" consisting of all clauses on the given vocabulary, $\mathcal{L}_T$ for the chosen target language, and $\overline{\mathcal{L}_T}$ (or $\mathcal{L}_R$) for $\mathcal{L} \setminus \mathcal{L}_T$, the complement of the target language. A theory is also a set of clauses. We assume basic familiarity with resolution inference procedures (including factoring), and associated concepts and terminology such as resolution trees, etc.

Definition 1 *A clause C subsumes a clause D iff $C\sigma \subseteq D$ for some substitution σ. C θ-subsumes D iff C subsumes D and C has no more literals than D.*

In the propositional case both notions of subsumption are identical to the subset relation. In FOL, if C subsumes D but does not θ-subsume it, then there exists a factor $C\sigma$ of C such that $C\sigma$ θ-subsumes D.

Definition 2 *A clausal language $\mathcal{L}_T$ is* closed under (θ-) subsumption *iff for every $C \in \mathcal{L}_T$, if a clause C' subsumes (θ-subsumes, respectively) C then $C' \in \mathcal{L}_T$.*

Definition 3 *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 $A \in \mathcal{L}_T$.*

We will also refer to closure under *ground* subsumption and *ground* resolution, meaning the restriction of the corresponding properties to the ground portion of the given language.

Definition 4 *A clause D is a* strict ground instance *of a clause C iff D is ground, $C\sigma = D$ for some substitution σ, and C and D contain the same number of predicate occurrences. C is said to be a strict generalization of D.*

Note that if D is a ground instance of C then D is a strict ground instance of either C or a factor of C. For example, let $C = P(x) \vee P(y)$. $P(a)$ is a (non-strict) ground instance of C, and a strict ground instance of a factor of C, namely $P(x)$.

Definition 5 *A clausal language $\mathcal{L}_T$ is* closed under strict ground instantiation *if $D \in \mathcal{L}_T$ whenever $C \in \mathcal{L}_T$ and D is a strict ground instance of C.*

Obviously, if $\mathcal{L}_T$ is closed under ground instantiation then $\mathcal{L}_R = \overline{\mathcal{L}_T}$ is closed under strict generalization of ground clauses. That is, if $D \in \mathcal{L}_R$ is a strict ground instance of C then $C \in \mathcal{L}_R$.

LUB approximations

LUB approximations were introduced in (Selman and Kautz 1991) for the Horn case, and then generalized in (Kautz and Selman 1991, Selman and Kautz 1995). 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 (or simply better suited to some specific task) target language $\mathcal{L}_T$. We will only be concerned with one type of bound, the $\mathcal{L}_T$-lowest upper bound ($\mathcal{L}_T$-LUB for short) Σ_{lub} of the input theory Σ. For reasons of

space, we focus only on those details that are technically important to the paper; additional motivation and applications of the LUB approach can be found in (Selman and Kautz 1991, Selman and Kautz 1995, del Val 1995). Let $Mod(\Gamma)$ denote the set of models of Γ, and let $\models$ denote the classical logical consequence relation. The $\mathcal{L}_T$-LUB of a theory Σ is the strongest theory of the language $\mathcal{L}_T$ entailed by Σ, that is:

Definition 6 *$\Sigma_{lub} \subseteq \mathcal{L}_T$ is an $\mathcal{L}_T$-*lowest upper bound (LUB) *of $\Sigma \subseteq \mathcal{L}$ iff $Mod(\Sigma) \subseteq Mod(\Sigma_{lub})$ (i.e. Σ_{lub} is an $\mathcal{L}_T$-upper bound of Σ) and for no $\Sigma' \subseteq \mathcal{L}_T$ it is the case that $Mod(\Sigma) \subseteq Mod(\Sigma') \subset Mod(\Sigma_{lub})$.*

The $\mathcal{L}_T$-LUB of a theory is equivalent to the conjunction of all $\mathcal{L}_T$-upper bounds; hence it always exists and is unique up to logical equivalence. Furthermore, when the query language is a subset of $\mathcal{L}_T$ then only the $\mathcal{L}_T$-LUB is needed, i.e. the $\mathcal{L}_T$-LUB is deductively complete with respect to $\mathcal{L}_T$-queries:

Theorem 1 *(del Val 1995, 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 $\Sigma_{lub} \models C$*

More generally, we can use the $\mathcal{L}_T$-LUB Σ_{lub} and the $\mathcal{L}_T$-GLB Σ_{glb} of Σ (GLBs are defined similarly) 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).

Computing ground LUBs

GLUB-1, the previously known algorithm for computing the $\mathcal{L}_T$-LUB of a ground (propositional) theory, is due to (Selman and Kautz 1991). The method consists in running a resolution procedure to exhaustion, under the restriction that at least one parent clause in each resolution step is *not* in the target language. We propose a new method, GLUB-2, which strengthens this requirement by disallowing resolutions between clauses both of which are not in $\mathcal{L}_T$. E.g. if $\mathcal{L}_T$ is the Horn language then the old algorithm requires that at least one parent clause is non-Horn, whereas the new algorithm requires that exactly one clause is non-Horn. Leaving aside details about deletion strategies (specifically deletion of subsumed and tautologous clauses), both algorithms can be succintly described as follows:

- **GLUB-1$(\Sigma, \mathcal{L}_T)$:** Compute the resolution closure of Σ under the restriction that *at least one parent clause in each resolution step is in $\mathcal{L}_R = \overline{\mathcal{L}_T}$*, storing $\mathcal{L}_T$-clauses in Σ^1_T and $\mathcal{L}_R$-clauses in Σ^1_R.
- **GLUB-2$(\Sigma, \mathcal{L}_T)$:** Compute the resolution closure of Σ under the restriction that *exactly one parent clause in each resolution step is in $\mathcal{L}_R = \overline{\mathcal{L}_T}$*, storing $\mathcal{L}_T$-clauses in Σ^2_T and $\mathcal{L}_R$-clauses in Σ^2_R.

The $\mathcal{L}_T$-LUB of Σ is given by the final value of Σ^i_T, hence Σ^1_T and Σ^2_T are equivalent for those target languages in which both algorithms are correct. GLUB-1 was proved correct for LUB generation for any target language closed under subsumption, and extended to arbitrary target languages, in (del Val 1995). GLUB-2, as discussed below, is correct if and only if the target language is closed under subsumption and its complement is closed under resolution. Only GLUB-1 has been proven compatible with deletion strategies such as deletion of subsumed and tautologous clauses.

Any resolution deduction satisfying the restriction of GLUB-1, is called an $\mathcal{L}_R$-resolution deduction (for the requirement that at least one parent clause is in the "restricted language" $\mathcal{L}_R$).[1] Any resolution deduction satisfying the restriction of GLUB-2 is called and $\mathcal{L}_R\mathcal{L}_T$-resolution deduction. We use the following notation for (propositional or first order) derivability relations, where Γ is a set of clauses and C a clause:

- $\Gamma \vdash C$ if there is a resolution deduction of a clause C' that subsumes C from Γ.
- $\Gamma \vdash_R C$ if there is an $\mathcal{L}_R$-resolution deduction of a clause C' that subsumes C from Γ.
- $\Gamma \vdash_{RT} C$ if there is an $\mathcal{L}_R\mathcal{L}_T$-resolution deduction of a clause C' that subsumes C from Γ.
- $\Gamma \Rightarrow C$ ($\Gamma \Rightarrow_\theta C$) if some clause of Γ subsumes (θ-subsumes, respectively) C.
- $\Gamma \vdash^* C$ if there is a resolution deduction of C (not just of a clause C' subsuming C) from Γ. Similarly $\Gamma \vdash^*_R C$, etc.

The key results regarding the deductive power of these algorithms in the ground case are as follows. Here Σ is a set of ground clauses, and C is a ground clause. Σ^i_j is the final value of the variable Σ^i_j after execution of the respective algorithm.

Lemma 2 *(del Val 1995, lemma 6) (GLUB-1) $\Sigma \vdash C$ iff either $\Sigma^1_T \vdash C$ or $\Sigma^1_R \Rightarrow C$.*

Lemma 3 *(GLUB-2) Suppose $\mathcal{L}_T$ is closed under subsumption and $\mathcal{L}_R = \overline{\mathcal{L}_T}$ is closed under resolution. $\Sigma \vdash^* C$ iff either $\Sigma^2_T \vdash^* C$ or $\Sigma^2_R \vdash^* C$.*

[1] This terminology is consistent with standard usage when $\mathcal{L}_R$ is replaced by the specific language name. Various common forms of resolution, such as unit resolution, input resolution, set of support resolution, and semantic resolution, are special cases of $\mathcal{L}_R$-resolution. Hence the results of this paper also provide a partial analysis of a whole family of important resolution procedures.

It is easy to show with lemma 3 that Σ_T^2 is the $\mathcal{L}_T$-LUB of Σ when the assumptions about the target language and its complement are satisfied. We defer the formal statement of this result to the first order case.

Apparently, the difference between both algorithms is captured by the replacement of $\Sigma_R^1 \models\!\!\Rightarrow C$ by $\Sigma_R^2 \vdash^* C$. For certain kinds of clauses (mostly those of $\mathcal{L}_R$, in particular the $\mathcal{L}_R$ prime implicates of Σ), GLUB-1 computes them all whereas GLUB-2 only gives us the means (that is, Σ_N^2) to derive them all. If we are only interested in $\mathcal{L}_T$-queries, on the other hand, then the extra effort invested by GLUB-1 is clearly superfluous. In particular, using the family of theories of (del Val 1995, corollary 12), on which GLUB-2 performs no resolutions at all, we can easily show:

Theorem 4 *GLUB-2 can produce exponential space and time savings by comparison with GLUB-1.*

First order compilation

We now proceed to lifting the correctness of both algorithms to the FOL case. GLUB-1 and GLUB-2 are essentially identical in the first order case, with an important qualification: Factoring is treated as a *separate* inference rule, which can be applied at any time to generate new clauses, but which is *not* regarded as part of the resolution rule. Thus, in the first order case, with $\mathcal{L}_T = Horn$, both GLUB-1 and GLUB-2 allow resolutions between e.g. a non-Horn clause and a Horn factor of a non-Horn clause, and forbid resolutions between a Horn clause and a Horn factor of a non-Horn clause.[2] Obviously, this implies that the subsumption criterion for deletion, if any, should be θ-subsumption, as otherwise all factors would be deleted.

Not every form of first order compilation is guaranteed to terminate on every instance, in particular forms of compilation which are complete for ($\mathcal{L}_T$-)consequence-finding. The first requirement in order to prove the correctness of the lifted algorithms is therefore to shift perspective from the possibly infinite sets Σ_j^i to a more computational view of *eventual derivability in finite time*, by restating lemmas 2 and 3 in terms of the deduction method used, $\mathcal{L}_R$- or $\mathcal{L}_R\mathcal{L}_T$-resolution. The next theorem does precisely this, and then lifts both kinds of deductions to the first order with an additional assumption about instantiation:

Theorem 5 *Suppose $\mathcal{L}_T$ and $\mathcal{L}_R = \overline{\mathcal{L}_T}$ are closed under strict ground instantiation. Then:*

1. *(GLUB-1)* $\Sigma \models C$ *iff:*

   (a) either *there exists a set of (target language) clauses* $\Gamma \subseteq \mathcal{L}_T$ *such that:*

   - $\Gamma \models C$;
   - *for every* $C_i \in \Gamma$, $\Sigma \vdash_R^* C_i$;

   (b) or $\Sigma \vdash_R C$.

2. *(GLUB-2) Suppose $\mathcal{L}_T$ is closed under ground subsumption and $\mathcal{L}_R$ is closed under ground resolution.* $\Sigma \vdash^* C$ *iff there exists a set of clauses* Γ *such that:*

   - *for every* $C_i \in \Gamma$, $\Sigma \vdash_{RT}^* C_i$;
   - $\Gamma \vdash^* C$;
   - *either* $\Gamma \subseteq \mathcal{L}_T$ *or* $\Gamma \subseteq \mathcal{L}_R$.

This theorem is compatible with a non-terminating inference procedure in the sense that it guarantees that whenever $\Sigma \models C$ we will eventually derive in finite time, with the appropriate form of resolution, either C or some finite set Γ with the required properties.

The ground version of theorem 5 is a direct consequence of lemmas 2 and 3. The theorem is obtained by lifting its ground version through a variant of a technique due to (Slagle *et al.* 1969). Using closure under strict instantiation of *both* sublanguages, we ensure that the lifted deductions are of the required kind.

The correctness of GLUB-1 and GLUB-2 for computing the $\mathcal{L}_T$-LUB in the first order case follows directly from theorem 5, interpreting the sets Σ_j^i infinitistically if needed.

Theorem 6 *(First order correctness of GLUB-1 and GLUB-2) Suppose $\mathcal{L}_T$ and $\mathcal{L}_R = \overline{\mathcal{L}_T}$ are closed under strict ground instantiation. Then:*

1. *(GLUB-1) If $\mathcal{L}_T$ is closed under θ-subsumption then Σ_T^1 is the $\mathcal{L}_T$-LUB of Σ.*

2. *(GLUB-2) If $\mathcal{L}_T$ is closed under θ-subsumption and $\mathcal{L}_R$ is closed under ground resolution, then Σ_T^2 is the $\mathcal{L}_T$-LUB of Σ.*

In particular, both Σ_T^1 and Σ_T^2 are sufficient to answer all $\mathcal{L}_T$-queries. Other properties of the GLUB algorithms can also be lifted easily. In particular:

Theorem 7 *Suppose $\mathcal{L}_T$ and $\mathcal{L}_R$ are closed under strict ground instantiation.*

1. *(GLUB-1) If $\mathcal{L}_T$ is closed under resolution and factoring*[3] *then* Σ_R^1 contains *all $\mathcal{L}_R$-prime implicates of Σ (i.e. all $\mathcal{L}_R$-consequences of Σ not properly θ-subsumed by some other consequence of Σ).*

2. *(GLUB-2) If $\mathcal{L}_T$ is closed under θ-subsumption, resolution, and factoring, and $\mathcal{L}_R$ is closed under ground resolution then* Σ_R^2 entails *(but does not necessarily contain) all $\mathcal{L}_R$-prime implicates of Σ.*

[2] We view the latter case simply as a resolution between two Horn clauses (one of which happens to be obtained by factoring from a non-Horn clause, but this is irrelevant).

[3] $\mathcal{L}_T$ is closed under factoring if every factor of an $\mathcal{L}_T$-clause is also in $\mathcal{L}_T$.

Target languages

The results of the previous section show that lifting the results on LUB approximations to the first order case is rather subtle, involving quite a few properties of languages. Fortunately, these turn out to be the "right" properties, in the sense that most interesting target languages which have the corresponding properties in the ground case also have them in their natural generalization to the first order. We briefly discuss some of these target languages next, noting first that all of them (and their complements) are closed under *strict* ground instantiation.[4]

1. Any subset of k-CNF (at most k literals per clause, for some fixed k) which is closed under θ-subsumption, e.g. k-Horn (Horn clauses with at most k literals). As in the ground case, we must use GLUB-1, since their complements are generally not closed under (ground) resolution. This choice of target language will tend to ensure small LUB approximations.

2. Horn and more generally k-quasi-Horn (clauses with at most k positive literals, for a fixed k) is closed under θ-subsumption (and for $k = 1$, under resolution and factoring). Furthermore, its complement is closed under *ground* resolution. Hence one can use GLUB-2, and in the Horn case also theorem 7.

3. The language $\mathcal{L}_T = \mathcal{L}_{\neg\exists l \in L}$ consisting of all clauses that do not contain instances of literals from a satisfiable set L of literals. E.g. L can be an interpretation, or, in diagnosis, the positive abnormality literals. See (del Val 1995, Inoue 1992) for applications of this kind of target language. $\mathcal{L}_{\neg\exists l \in L}$ is closed under θ-subsumption and resolution, and its complement is closed under resolution as L cannot contain complementary literals. Hence GLUB-2 can be used and theorem 7 can also be applied.

The combinations of properties which are required for each algorithm to be correct in the first order case are in a sense quite fortunate. For example, non-Horn is not closed under ground instantiation ($Q(a)$ is a ground instance of $Q(x) \vee Q(y)$) but it is closed under *strict* ground instantiation. Neither the k-CNF languages nor the k-quasi-Horn languages are closed under (standard) subsumption (e.g. the non-Horn clause $C = P(x,y) \vee P(y,z) \vee \neg Q(x,z)$ subsumes the Horn clause $D = P(x,x) \vee \neg Q(x,x)$) yet they are all closed under θ-subsumption. Finally, of essential importance for GLUB-2, we only need closure under *ground* resolution of the complement language in order for the first order GLUB-2 to work. For example, the complement of k-quasi-Horn is closed under ground resolution, but not under general resolution (e.g. the two non-Horn clauses $p(x,y) \vee p(y,z) \vee \neg q(x,y,z)$ and $q(v,v,v) \vee p(v,v)$ have as resolvent the Horn clause $p(v,v)$). In all these cases, the corresponding version for the lifted algorithm of any of the properties required for the ground case behaves just perfectly for the lifted versions of the languages of interest.

[4] The only interesting languages that seem to be ruled out by the instantiation conditions are *ground* target languages, as their complements are not closed under ground instantiation.

Issues in first order compilation

As mentioned, the nature of first order inference raises a number of significant and challenging issues for knowledge compilation. We briefly discuss them next:

The complexity of inference and the choice of target language. While certain properties of languages can be lifted relatively easily to FOL, the complexity of inference is not one of them. In particular, tractable target languages are much harder to find. The non-ground Horn language (with function symbols) is in essence as expressive as full first order predicate calculus (Hodges 1993, section 10), hence entailment and satisfiability when using first order Horn LUBS are at best semidecidable; whereas they are polynomially decidable in the ground case. Recall however that the choice of target language can also be guided by the desire to obtain a theory which is complete with respect to $\mathcal{L}_T$-queries, while discarding other information which is "irrelevant" (for a given task). Once the $\mathcal{L}_T$-LUB is computed, ensuring this degree of relative completeness, it can be further compiled by some other method.

Termination. Even for decidable subsets of FOL, compilation algorithms are not always guaranteed to terminate. Consider the following theories:

Σ_1 : $\neg e(x1,x2) \vee p(x1,x2)$;
$\neg e(y1,y2) \vee \neg p(y2,y3) \vee p(y1,y3)$.

Σ_2 : $\neg e(x1,x2) \vee p(x1,x2)$;
$\neg e(y1,y2) \vee \neg p(y2,y3) \vee p(y1,y3) \vee q(y1,y3)$.

Σ_3 : $\neg e(x,y) \vee \neg f(x,y)$;
$e(x,y) \vee f(x,y)$;
$\neg e(x1,x2) \vee p(x1,x2)$;
$\neg e(y1,y2) \vee \neg p(y2,y3) \vee p(y1,y3)$.

For simplicity, let $\mathcal{L}_T$ be the Horn language. Then:

1. Any approach which is complete for consequence-finding, such as (Minicozzi and Reiter 1972, Inoue 1992) fails to terminate with either theory, since they all have an infinite number of prime consequences. Both GLUB-1 and GLUB-2 terminate with Σ_1, without performing any resolutions at all.

2. GLUB-1 fails to terminate on Σ_2, which has an infinite number of non-Horn unsubsumed consequences. GLUB-2 terminates almost immediately.

3. GLUB-1 and GLUB-2 fail to terminate on Σ_3, generating an infinite sequence of clauses of the form:

$$\neg e(z1, z2) \vee \neg e(z2, z3) \vee \ldots \vee \neg e(z_{n-1}, z_n) \vee p(z_1, z_n);$$
and
$$f(z1, z2) \vee \neg e(z2, z3) \vee \ldots \vee \neg e(z_{n-1}, z_n) \vee p(z_1, z_n).$$

Note that the corresponding Horn LUBs are in all cases finitely axiomatizable, so non-termination cannot be justified on these grounds. Furthermore, entailment with respect to any of these theories is decidable, as they have a finite Herbrand universe, so the use of a decidable subset of FOL is not sufficient to guarantee termination.

Fairness and completeness. While it would certainly be desirable to have good termination criteria, there is neverthless an important sense in which the lack of termination guarnatees does not affect the completeness of the compilation procedures, any more than it affects the completeness for consequence finding of a given resolution procedure. What is needed in both cases is a *fair* control strategy, which ensures that any derivable clause is eventually derived in finite time. In the case of procedures which are complete for consequence-finding, fairness requires that every two clauses which are queued for resolution are eventually resolved together. In the case of compilation procedures, we need an additional dimension of fairness. Namely, the compilation procedure must be interleaved or executed in parallel with the query answering procedure. For any given $C \in \mathcal{L}_T$ the algorithms will eventually compute a finite set Γ with the desired properties (being an upper bound of Σ and entailing C). Since we interleave compilation and query answering (which is also assumed to use a fair control strategy), we will eventually answer that C does follow from Σ, in finite time. If C does not follow from Σ then we may not be able to ever tell, however. Similar comments apply to other approaches to compilation. This interleaving strategy is therefore essential in the absence of termination guarantees, in contrast with the usual view of compilation according to which query answering only begins after compilation is completed.

Discussion

We have provided a new ground algorithm for computing LUB approximations, and shown that both the new and the old algorithm can be lifted under very general conditions to the first order case. Because FOL is much more concise (let alone expressive) than propositional logic, the feasibility of any approach to knowledge compilation depends essentially on its applicability to the first order case. While propositional methods are often useful and often used in practice, through instantiation in finite domains, we believe a more practical and more principled approach requires us to address the first order case head on. We hope the techniques of this paper can be used for other approaches to knowledge compilation, most of which have only focused on the propositional case. At the same time, there are a number of open issues which arise in first order compilation and which need to be addressed if the general idea of knowledge compilation is to be applied to the first order calculus in a well-behaved manner.

References

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.

Alvaro del Val. An analysis of approximate knowledge compilation. In *IJCAI'95, Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 830–836, 1995.

Wilfred Hodges. Logical features of Horn clauses. In Dov M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 449–503. Oxford University Press, 1993.

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

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

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

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.

Pierre Marquis. Knowledge compilation using theory prime implicates. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 837–843, 1995.

Eliana Minicozzi and Raymond Reiter. A note on linear resolution strategies in consequence-finding. *Artificial Intelligence*, 3:175–180, 1972.

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

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

J. R. Slagle, C. L. Chang, and R. C. T. Lee. Completeness theorems for semantic resolution in consequence finding. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1969.