Computing Strongest Necessary and Weakest Sufficient Conditions of First-Order Formulas

Patrick Doherty*
Dept. of Computer Science
Linköping University
S-581 83 Linköping, Sweden
email: patdo@ida.liu.se

Witold Łukaszewicz†
Andrzej Szałas†
Dept. of Computer Science, Linköping University
and College of Economics and Computer Science
TWP, Olsztyn, Poland
email: witlu,andsz@ida.liu.se

Abstract

A technique is proposed for computing the weakest sufficient (wsc) and strongest necessary (snc) conditions for formulas in an expressive fragment of first-order logic using quantifier elimination techniques. The efficacy of the approach is demonstrated by using the techniques to compute snc's and wsc's for use in agent communication applications, theory approximation and generation of abductive hypotheses. Additionally, we generalize recent results involving the generation of successor state axioms in the propositional situation calculus via snc's to the first-order case. Subsumption results for existing approaches to this problem and a re-interpretation of the concept of *forgetting* as a process of quantifier elimination are also provided.

In Proceedings of the 17th Int'l Joint Conference on Artificial Intelligence, August 4th-10th, 2001, Seattle, Washington, USA (IJCAI-2001).

1 Introduction

In [Lin, 2000][1], Lin proposes the notion of weakest sufficient and strongest necessary conditions for a proposition q under a propositional theory T, where the techniques for generating the resulting formulas may be parameterized to contain only a restricted subset of the propositional variables in T. In addition, he investigates a number of methods for automatically generating snc's and wsc's, several based on the generation of prime implicates and a preferred method for computing snc's and wsc's based on the notion of *forgetting* [Lin and Reiter, 1994]. Snc's and wsc's have many potential uses and applications ranging from generation of abductive hypotheses to approximation of theories. In fact, special cases of snc's and wsc's, strongest postconditions and weakest preconditions, have had widespread usage as a basis for programming language semantics [Dijkstra, 1976].

Weakest sufficient and strongest necessary conditions for propositional formulas are related to prime implicants and implicates, respectively. Classically, a prime implicant of a formula α is a minimal satisfiable term logically implying α and a prime implicate is a minimal satisfiable clause which is logically implied by α. The relation between prime implicants/implicates and wsc's/snc's is used by Lin in his investigation of methods for automatically generating wsc's and snc's. Generating prime implicants and implicates for propositional formulas is intractable and in the general case, the same applies for wsc's and snc's.

Lin's results apply to the propositional case and his algorithms for automatically generating snc's and wsc's are empirically tested. For the propositional case, we propose a different method for computing snc's and wsc's that is based on second-order quantifier elimination techniques and has the following advantages:

- The general method applies to the full propositional language and snc's and wsc's are generated directly for arbitrary formulas rather than propositional atoms.
- When applying second-order quantifier elimination, one substantially simplifies the propositional case considered by Lin and often gets a more efficient computation method.
- A non-trivial fragment of the propositional language is isolated where snc's and wsc's for formulas in this fragment can be generated efficiently and are guaranteed to be so.
- The quantifier elimination algorithms which provide the basis for automatically generating snc's and wsc's for arbitrary propositional formulas are implemented.

One of the most interesting and potentially fruitful open problems regarding snc's and wsc's is generalization to the first-order case and developing associated methods for automatically computing snc's and wsc's. In [Lin, 2000], Lin states,

> There are several directions for future work. One of them is to extend the results here to the first-order case. This can be a difficult task. For instance, a result in [Lin

*Supported in part by the WITAS project grant under the Wallenberg Foundation, Sweden.

†Supported in part by the WITAS project grant under the Wallenberg Foundation, Sweden and KBN grant 8 T11C 009 19.

[1]This paper received the best paper award at the KR'00 Conference, Breckenridge, Colorado, 2000.

and Reiter, 1997] shows that forgetting in the first-order case cannot in general be expressible in first-order logic. As a consequence, we expect that strongest necessary conditions of a proposition under a first-order theory cannot in general be expressible in first-order logic either. It seems that the best hope for dealing with the first -order case is to first reduce it to the propositional case, and then try to learn a first-order description from a set of propositional ones.

With Lin, we agree that the task is difficult. We also agree that in the general case snc's and wsc's for a proposition or first-order formula under a first-order theory is not always expressible in first-order logic. In fact, the techniques we propose provide a great deal of insight into why this is the case and in addition define a non-trivial fragment of first-order logic where snc's and wsc's are guaranteed to be expressible in first-order logic.

Rather than using indirect techniques to reduce a first-order case to the propositional case and then try to learn a first-order description from a set of propositional ones as Lin suggests, we propose a more direct method and provide techniques for the automatic generation of snc's and wsc's for a first-order fragment. Given a theory T and a formula α in this fragment, we simply append appropriate quantifiers over relational variables in $T \wedge \alpha$ (or $T \rightarrow \alpha$) and use second-order quantifier elimination techniques to reduce the second-order formula to a logically equivalent first-order formula representing the snc or the wsc for α under T. Complexity results are provided for this fragment, but the method works for full first-order logic. In this case, depending on the nature of T and α, the technique may return a logically equivalent first-order *or* fixpoint formula, or terminate with failure, not always because there is not a reduction, but simply because the elimination algorithm can not find a reduction.

We compute these conditions using extensions of results described in the work of [Doherty *et al.*, 1997; 1998; Nonnengart and Szalas, 1998]. For a survey of these and other quantifier elimination techniques, see also [Nonnengart *et al.*, 1999].

1.1 Weakest Sufficient and Strongest Necessary Conditions

In the following, we will be dealing with the predicate calculus with equality, i.e. we assume that the equality, $=$, is always a logical symbol. The following definitions describe the necessary and sufficient conditions of a formula α relativized to a subset P of relation symbols under a theory T.

Definition 1.1 By *a necessary condition of a formula α on the set of relation symbols P under theory T* we shall understand any formula ϕ containing only symbols in P such that $T \models \alpha \rightarrow \phi$. It is a *strongest necessary condition*, denoted by $\mathsf{SNC}(\alpha; T; P)$ if, additionally, for any necessary condition ψ of α on P under T, we have that $T \models \phi \rightarrow \psi$. ■

Definition 1.2 By *a sufficient condition of a formula α on the set of relation symbols P under theory T* we shall understand any formula ϕ containing only symbols in P such that $T \models \phi \rightarrow \alpha$. It is a *weakest sufficient condition*, denoted by $\mathsf{WSC}(\alpha; T; P)$ if, additionally, for any sufficient condition ψ of α on P under T, we have that $T \models \psi \rightarrow \phi$. ■

The set P in Definitions 1.1 and 1.2 is referred to as the *target language*.

To provide some intuition as to how these definitions can be used, consider the theory

$$T = \{\forall x.[HasWheels(x) \rightarrow CanMove(x)], \\ \forall x.[Car(x) \rightarrow HasWheels(x)]\},$$

and the formula $\alpha = \forall x.CanMove(x)$. Clearly $T \not\models \alpha$. Quite often, it is useful to hypothesize a preferred explanation ϕ for α under a theory T where $T \wedge \phi \models \alpha$, ϕ is minimal in the sense of not being overly specific and where the explanation is constrained to a particular subset P of symbols in the vocabulary. Clearly, the weakest sufficient condition ϕ for the formula α on P under T provides the basis for a minimal preferred explanation of α where $T \models \phi \rightarrow \alpha$. In the case of $P = \{HasWheels\}$, the weakest sufficient condition would be $\phi = \forall x.HasWheels(x)$, and in the case of $P = \{HasWheels, Car\}$ the wsc would be $\phi = \forall x.HasWheels(x)$. Generating abductive hypotheses is just one application of wsc's. There are many other applications which require generation of wsc's or snc's, several of which are described in section 5.

1.2 Paper Structure

In section 2, we begin with preliminaries and state the theorem which provides the basis for second-order quantifier elimination techniques. In section 3, we define snc's and wsc's for propositional formulas under propositional theories as second-order formulas with quantification over propositional symbols, show how the elimination techniques are applied and provide complexity results for the technique. In section 4, we generalize to the first-order case using primarily the same techniques, but with quantification over relational symbols. In section 5, we demonstrate both the use of snc's and wsc's in addition to the reduction techniques by providing examples from a number of potentially interesting application areas such as agent communication languages, theory approximation, generation of abductive hypotheses and generation of successor state axioms in the situation calculus. In section 6, we relate the proposed techniques to the notion of *forgetting* which serves as a basis for Lin's techniques and in so doing, prove subsumption results. In section 7, we conclude with a discussion about these results.

2 Preliminaries

The following Theorem 2.2 has been proved in [Nonnengart and Szalas, 1998]. It allows us to eliminate second-order quantifiers from formulas which are in the form appearing on the left-hand side of the equivalences (1), (2). Such formulas are called *semi-Horn formulas* w.r.t. the relational variable Φ - see also [Doherty *et al.*, 1996]. Observe, that in the context of databases one remains in the tractable framework, since fixpoint formulas over finite domains are computable in polynomial time (and space) - see e.g. [Abiteboul *et al.*, 1996; Ebbinghaus and Flum, 1995]. (Of course, the approach is applicable to areas other than databases, too).

Notation 2.1 Let e, t be any expressions and s any subexpression of e. By $e(s := t)$ we shall mean the expression

obtained from e by substituting each occurrence of s by t. $\mu\Phi.A(\Phi)$ is the least fixpoint operator and $\nu\Phi.A(\Phi)$ is defined as $\neg\mu\neg\Phi.\neg A(\Phi)$.

Let $A(\bar{x})$ be a formula with free variables $\bar{x}$. Then by $A(\bar{x})[\bar{a}]$ we shall mean the application of $A(\bar{x})$ to arguments $\bar{a}$. ■

Theorem 2.2 Assume that all occurrences of the predicate variable Φ in the formula B bind only variables and that formula A is positive w.r.t. Φ.

- if B is negative w.r.t. Φ then

$$\exists\Phi\forall\bar{y}\,[A(\Phi) \rightarrow \Phi(\bar{y})] \wedge [B(\neg\Phi)] \;\equiv\; B[\Phi(\bar{t}) := \mu\Phi(\bar{y}).A(\Phi)[\bar{t}]] \qquad (1)$$

- if B is positive w.r.t. Φ then

$$\exists\Phi\forall\bar{y}[\Phi(\bar{y}) \rightarrow A(\Phi)] \wedge [B(\Phi)] \;\equiv\; B[\Phi(\bar{t}) := \nu\Phi(\bar{y}).A(\Phi)[\bar{t}]]. \qquad (2)$$

■

Example 2.3 Consider the following second-order formula:

$$\exists\Phi\forall x\forall y[(S(x,y) \vee \Phi(y,x)) \rightarrow \Phi(x,y)] \wedge[\neg\Phi(a,b) \vee \forall z(\neg\Phi(a,z))]] \qquad (3)$$

According to Theorem 2.2(1), formula (3) is equivalent to:

$$\neg\mu\Phi(x,y).(S(x,y) \vee \Phi(y,x))[a,b] \vee \forall z(\neg\mu\Phi(x,y).(S(x,y) \vee \Phi(y,x))[a,z]). \qquad (4)$$

■

Observe that, whenever formula A in Theorem 2.2 does not contain Φ, the resulting formula is easily reducible to a first-order formula, as in this case both $\mu\Phi(\bar{y}).A$ and $\nu\Phi(\bar{y}).A$ are equivalent to A. In fact, this case is equivalent to the lemma of Ackermann (see e.g. [Doherty *et al.*, 1997]). Semi-Horn formulas of the form (1) and (2), where A does not contain Φ, are called *non-recursive semi-Horn formulas*.

3 The Propositional Case

In this section, we define snc's and wsc's for propositional formulas under propositional theories as second-order formulas with quantification over propositional symbols, show how the elimination techniques are applied and provide complexity results for the technique. We start with the following lemma.

Lemma 3.1 For any formula α, any set of propositional symbols P and theory Th:

1. the strongest necessary condition $\mathsf{SNC}(\alpha; Th; P)$ is defined by $\exists\bar{q}.[Th \wedge \alpha]$,
2. the weakest sufficient condition $\mathsf{WSC}(\alpha; Th; P)$ is defined by $\forall\bar{q}.[Th \rightarrow \alpha]$,

where $\bar{q}$ consists of all propositions appearing in Th and α but not in P.

Proof The proof of the lemma for both the strongest necessary and weakest sufficient conditions are similar, but we provide both for clarity.

By definition, any necessary condition ϕ for α satisfies $Th \models \alpha \rightarrow \phi$, i.e. by the deduction theorem for propositional calculus, also $\models Th \rightarrow (\alpha \rightarrow \phi)$, i.e. $\models (Th \wedge \alpha) \rightarrow \phi$. Thus also $\models \forall\bar{q}.[(Th \wedge \alpha) \rightarrow \phi]$. Since ϕ is required not to contain symbols from $\bar{q}$, we have

$$\models [\exists\bar{q}.(Th \wedge \alpha)] \rightarrow \phi. \qquad (5)$$

On the other hand, the minimal ϕ satisfying (5) is given by equivalence

$$[\exists\bar{q}.(Th \wedge \alpha)] \equiv \phi.$$

This proves Lemma 3.1.1.

By definition, any sufficient condition ϕ for α satisfies $Th \models \phi \rightarrow \alpha$, i.e. by the deduction theorem for propositional calculus, also $\models Th \rightarrow (\phi \rightarrow \alpha)$, i.e. $\models (Th \wedge \phi) \rightarrow \alpha$. Thus also $\models \forall\bar{q}.[(Th \wedge \phi) \rightarrow \alpha]$ which is equivalent to $\models \forall\bar{q}.[(Th \wedge \neg\alpha) \rightarrow \neg\phi]$.

Since ϕ is required not to contain symbols from $\bar{q}$, we have

$$\models [\exists\bar{q}.(Th \wedge \neg\alpha)] \rightarrow \neg\phi. \qquad (6)$$

Maximizing ϕ is the same as minimizing $\neg\phi$. On the other hand, the maximal ϕ satisfying (6) is given by equivalence

$$[\exists\bar{q}.(Th \wedge \neg\alpha)] \equiv \neg\phi.$$

which is equivalent to

$$\neg[\exists\bar{q}.(Th \wedge \neg\alpha)] \equiv \phi.$$

which is equivalent to

$$[\forall\bar{q}.(Th \rightarrow \alpha)] \equiv \phi.$$

This proves Lemma 3.1.2. ■

The quantifiers over propositions can be automatically eliminated using the DLS algorithm (see [Doherty *et al.*, 1997]). For instance, all eliminations in Example 3.3 can be done using the algorithm. Theorem 2.2 reduces in the propositional case to Proposition 3.2. It is worth emphasizing here that propositional fixpoint formulas are equivalent to propositional formulas. [2]

Proposition 3.2 Assume that the propositional formula A is positive w.r.t. proposition p.

- if the propositional formula B is negative w.r.t. p then

$$\exists p.\,[A(p) \rightarrow p] \wedge [B(\neg p)] \;\equiv\; B[p := \mu p.A(p)] \qquad (7)$$

- if B is positive w.r.t. p then

$$\exists p.[p \rightarrow A(p)] \wedge [B(p)] \;\equiv\; B[p := \nu p.A(p)]. \qquad (8)$$

■

[2] In the first iteration towards the fixpoint, one replaces p in A with false. In the next disjunct, p in A is replaced by this result. The fixpoint, a propositional formula, is then always reached in at most two iterations.

Observe that in the case when an input formula is a conjunction of propositional semi-Horn formulas of the form in the *lhs* of (7) or a conjunction of formulas of the form in the *lhs* of (8), the length of the resulting formula is, in the worst case, $O(n^2)$, where n is the size of the input formula. Otherwise the result might be of exponential length, as in the case of the algorithm given in [Lin, 2000].

Example 3.3 Consider the following examples of [Lin, 2000].

1. $T_1 = \{q \rightarrow (p_1 \wedge p_2)\}$. Now, according to Lemma 3.1, $\mathsf{SNC}(q; T_1; \{p_1, p_2\})$ is defined by formula $\exists q.[(q \rightarrow (p_1 \wedge p_2)) \wedge q]$, which, according to Proposition 3.2, is logically equivalent to $(p_1 \wedge p_2)$.

   Condition $\mathsf{SNC}(q; T_1; \{p_1\})$ is defined by formula $\exists q \exists p_2.[(q \rightarrow (p_1 \wedge p_2)) \wedge q]$, which, according to Proposition 3.2, is logically equivalent to p_1 (observe that p_2 is equivalent to the semi-Horn formula $\top \rightarrow p_2$).

2. $T_2 = \{q \rightarrow (p_1 \vee p_2)\}$. We have that $\mathsf{SNC}(q; T_2; \{p_1, p_2\})$ is defined by the formula $\exists q.[(q \rightarrow (p_1 \vee p_2)) \wedge q]$, which, according to Proposition 3.2, is logically equivalent to $(p_1 \vee p_2)$.

   Condition $\mathsf{SNC}(q; T_2; \{p_1\})$ is defined by the formula $\exists q \exists p_2.[(q \rightarrow (p_1 \vee p_2)) \wedge q]$, which is logically equivalent to $\top$.

3. $T_3 = \{(p \wedge q) \rightarrow s\}$. The formula $\mathsf{SNC}(p \wedge q; T_3; \{s\})$ is equivalent to $\exists p, q.[((p \wedge q) \rightarrow s) \wedge (p \wedge q)]$, which, according to Proposition 3.2, is logically equivalent to s.

Observe that we work with formulas more directly than proposed in Lin's approach, where a new proposition has to be introduced together with an additional condition that the new proposition is equivalent to the formula in question.

■

In summary, propositional snc's or wsc's can be generated for any propositional formula and theory. In the case that the conjunction of both is in semi-Horn form, this can be done more efficiently. These results subsume those of Lin [Lin, 2000] in the sense that the full propositional language is covered and we work directly with propositional formulas rather than propositional atoms.

4 The First-Order Case

In this section, we generalize the results in section 3 to the first-order case using primarily the same techniques, but with quantification over relational symbols. The following lemma can be proved similarly to Lemma 3.1. The deduction theorem for first-order logic is applicable, since the theories are assumed to be closed.

Lemma 4.1 For any formula α, any set of relation symbols P and a closed[3] theory Th:

1. the strongest necessary condition $\mathsf{SNC}(\alpha; Th; P)$ is defined by $\exists \bar{\Phi}.[Th \wedge \alpha]$,
2. the weakest sufficient condition $\mathsf{WSC}(\alpha; Th; P)$ is defined by $\forall \bar{\Phi}.[Th \rightarrow \alpha]$,

where $\bar{\Phi}$ consists of all relation symbols appearing in Th and α but not in P. ■

Observe that a second-order quantifier over the relational variable Φ can be eliminated from any semi-Horn formula w.r.t. Φ. In such a case the resulting formula is a fixpoint formula. If the formula is non-recursive, then the resulting formula is a first-order formula. The input formula can also be a conjunction of semi-Horn formulas of the form (1) or a conjunction of semi-Horn formulas of the form (2). On the other hand, one should be aware that in other cases the reduction is not guaranteed. Thus the elimination of second-order quantifiers is guaranteed for any formula of the form $\exists \bar{\Phi}.[Th \wedge \alpha]$, where $Th \wedge \alpha$ is a conjunction of semi-Horn formulas w.r.t. all relational variables in $\bar{\Phi}$.[4] Observe also, that in the case when an input formula is a conjunction of semi-Horn formulas of the form (1) or a conjunction of formulas of the form (2), the length of the resulting formula is, in the worst case, $O(n^2)$, where n is the size of the input formula.

Example 4.2 Consider the following examples

1. $T_4 = \{\forall x.[Ab(x) \rightarrow (Bird(x) \wedge \neg Flies(x))]\}$. Consider the strongest necessary condition $\mathsf{SNC}(Ab(z); T_4; \{Bird, Flies\})$. According to Lemma 4.1, it is equivalent to

$$\exists Ab.[\forall x.(Ab(x) \rightarrow (Bird(x) \wedge \neg Flies(x))) \wedge Ab(z)]. \quad (9)$$

   By Lemma 2.2, formula (9) is equivalent to $(Bird(z) \wedge \neg Flies(z))$.

2. $T_5 = \{\forall x.[Parent(x) \rightarrow \exists z.(Father(x, z) \vee Mother(x, z))]\}$. Consider the strongest necessary condition $\mathsf{SNC}(Parent(y); T_5; \{Mother\})$. According to Lemma 4.1, it is equivalent to

$$\exists Parent, Father.[\forall x.(Parent(x) \rightarrow \exists z.(Father(x, z) \vee Mother(x, z)) \wedge Parent(y)]. \quad (10)$$

   In this case, formula (10) is not in the form required in Lemma 2.2, but the DLS algorithm eliminates the second-order quantifiers and results in the equivalent formula $\top$, which is the required strongest necessary condition. Consider now $\mathsf{SNC}(Parent(y) \wedge \forall u, v.(\neg Father(u, v)); T_5; \{Mother\})$. It is equivalent to

$$\exists Parent, Father.[\forall x.(Parent(x) \rightarrow \exists z.(Father(x, z) \vee Mother(x, z)) \wedge Parent(y) \wedge \forall u, v.(\neg Father(u, v))], \quad (11)$$

   i.e. after eliminating second-order quantifiers, to $\exists z.Mother(y, z)$.

■

[3] In fact, it suffices to assume that the set of free variables of Th is disjoint from the set of free variables of α.

[4] For universal quantification, $\forall \bar{\Phi}.A$, one simply negates the formula $(\exists \bar{\Phi}.\neg A)$, and assuming $\neg A$ can be put in to semi-Horn form, one eliminates the existential quantifiers and negates the result.

In summary, for the non-recursive semi-Horn fragment of first-order logic, the snc or wsc for a formula α and theory T is guaranteed to be reducible to compact first-order formulas. For the recursive case, the snc's and wsc's are guaranteed to be reducible to fixpoint formulas. In the context of databases, this case is still tractable. The techniques may still be used for the full first-order case, but neither reduction nor complexity results are guaranteed, although the algorithm will always terminate.

5 Applications

In this section, we demonstrate the use of the techniques by applying them to a number of potentially useful application areas.

5.1 Communicating Agents

Agents communicating, e.g. via the Internet, have to use the same language to understand each other. This is similar or related to computing interpolants.

Assume an agent A wants to ask a query Q to agent B. Suppose the query can be asked using terms $\bar{R}, \bar{S}$ such that the terms from $\bar{S}$ are unknown for agent B. Let $T(\bar{R}, \bar{S})$ be a theory describing some relationships between $\bar{R}$ and $\bar{S}$. It is then natural for agent A to first compute the strongest necessary condition $\mathsf{SNC}(Q; T(\bar{R}, \bar{S}); \bar{R})$ with the target language restricted to $\bar{R}$ and then to replace the original query by the computed condition. The new query might not be as good as the previous one, but is the best that can be asked. The following example illustrates the idea.

Example 5.1 Assume an agent A wants to select from a database all persons x such that $High(x) \wedge Silny(x)$ holds. Assume further, that both agents know the terms $High$ and $Sound$. Unfortunately, the database agent does not know the term $Silny$.[5] Suppose, further that A lives in a world in which the condition $\forall y.[Silny(y) \rightarrow Sound(y)]$ holds. It is then natural for A to consider

$$\mathsf{SNC}(High(x) \wedge Silny(x); \forall y.[Silny(y) \rightarrow Sound(y)]; \{High, Sound\})$$

to be the best query that can be asked. According to Lemma 4.1 this condition is equivalent to:

$$\exists Silny.[\forall y.[Silny(y) \rightarrow Sound(y)] \wedge High(x) \wedge Silny(x),$$

which, by a simple application of Theorem 2.2, is equivalent to $High(x) \wedge Sound(x)$. ■

5.2 Theory Approximation

The concept of approximating more complex theories by simpler theories has been studied in [Kautz and Selman, 1996; Cadoli, 1995], mainly in the context of approximating arbitrary propositional theories by propositional Horn clauses. The concept of approximate theories is also discussed in [McCarthy, 2000]. Now, observe that strongest necessary and weakest sufficient conditions provide us with approximations of theories expressed in a richer language by theories expressed in a simpler language.

The approach by Lin in [Lin, 2000] allows one only to approximate simple concepts on the propositional level. The generalization we introduce allows us to approximate any finite propositional and first-order theory which is semi-Horn w.r.t. the eliminated propositions or relational symbols.

In the following example, considered in [Kautz and Selman, 1996] approximating general clauses by Horn clauses results in the exponential blow up of the number of clauses. We shall show, that the use of the notion of strongest necessary condition can substantially reduce the complexity of reasoning.

Example 5.2 In [Kautz and Selman, 1996] the following clauses, denoted by T, are considered:

$$(CompSci \wedge Phil \wedge Psych) \rightarrow CogSci \quad (12)$$
$$ReadsMcCarthy \rightarrow (CompSci \vee CogSci) \quad (13)$$
$$ReadsDennett \rightarrow (Phil \vee CogSci) \quad (14)$$
$$ReadsKosslyn \rightarrow (Psych \vee CogSci) \quad (15)$$

and reasoning with this theory was found to be quite complicated. On the other hand, one would like to check, for instance, whether a computer scientist who reads Dennett and Kosslyn is also a cognitive scientist. Reasoning by cases, suggested in [Kautz and Selman, 1996], shows that this is the case. One can, however, substantially reduce the theory and make the reasoning more efficient. In the first step one can notice that notions $Phil$ and $Psych$ are not in the considered claim, thus might appear redundant in the reasoning process. On the other hand, these notions appear in disjunctions in clauses (14) and (15). We then consider

$$\mathsf{SNC}(CompSci \wedge ReadsDennett \wedge ReadsKosslyn; T; -\{Phil, Psych\}), \quad (16)$$

where $-\{Phil, Psych\}$ denotes all symbols in the language, other than $Phil$ and $Psych$. After some simple calculations one obtains the following formula equivalent to (16):

$$\begin{aligned} (13) \quad \wedge \quad & [CompSci \wedge ReadsDennett \wedge ReadsKosslyn] \\ \wedge \quad & [(CompSci \wedge (ReadsDennett \wedge \neg CogSci) \\ \wedge \quad & (ReadsKosslyn \wedge \neg CogSci)) \rightarrow CogSci] \end{aligned} \quad (17)$$

which easily reduces to

$$\begin{aligned} &(13) \wedge CompSci \wedge ReadsDennett \wedge \\ &ReadsKosslyn \wedge (\neg CogSci \rightarrow CogSci). \end{aligned} \quad (18)$$

Thus the strongest necessary condition for the formula

$$CompSci \wedge ReadsDennett \wedge ReadsKosslyn$$

implies $CogSci$ and, consequently, the formula also implies $CogSci$.

Assume that one wants to calculate the weakest sufficient condition of being a computer scientist in terms of $\{ReadsDennett, ReadsKosslyn, ReadsMcCarthy, CogSci\}$. We then consider

$$\mathsf{WSC}(CompSci; T; -\{Phil, Psych, CompSci\}). \quad (19)$$

[5] In Polish "Silny" means "Strong", but the database agent does not know the Polish language.

After eliminating quantifiers over $Phil, Psych, CompSci$ from the second-order formulation of the wsc, one obtains the following formula equivalent to (19):

$$ReadsMcCarthy \wedge \neg CogSci.$$

Thus the weakest condition that, together with theory T, guarantees that a person is a computer scientist is that the person reads McCarthy and is not a cognitive scientist. ■

5.3 Abduction

The weakest sufficient condition corresponds to a weakest abduction, as noticed in [Lin, 2000].

Example 5.3 Consider theory

$$T = \{\forall x.[HasWheels(x) \rightarrow CanMove(x)], \\ \forall x.[Car(x) \rightarrow HasWheels(x)]\}.$$

Assume one wants to check whether an object can move. There are three interesting cases:

1. to assume that the target language is $\{HasWheels\}$ and consider

   $$\mathsf{WSC}(CanMove(x); T; \{HasWheels\}),$$

   which is equivalent to

   $$\forall CanMove, Car.[\bigwedge T \rightarrow CanMove(x)]$$

2. to assume that the target language is $\{Car\}$ and consider

   $$\mathsf{WSC}(CanMove(x); T; \{Car\}),$$

   which is equivalent to

   $$\forall HasWheels, Car.[\bigwedge T \rightarrow CanMove(x)]$$

3. to assume that the target language is $\{HasWheels, Car\}$ and consider

   $$\mathsf{WSC}(CanMove(x); T; \{HasWheels, Car\}),$$

   which is equivalent to

   $$\forall CanMove.[\bigwedge T \rightarrow CanMove(x)].$$

After eliminating second-order quantifiers we obtain the following results:

1. $\mathsf{WSC}(CanMove(x); T; \{HasWheels\}) \equiv HasWheels(x)$
2. $\mathsf{WSC}(CanMove(x); T; \{Car\}) \equiv Car(x)$
3. $\mathsf{WSC}(CanMove(x); T; \{HasWheels, Car\}) \equiv \forall x.[Car(x) \rightarrow HasWheels(x)] \rightarrow HasWheels(x)$.

The first two conditions are rather obvious. The third one might seem a bit strange, but observe that $\forall x.[Car(x) \rightarrow HasWheels(x)]$ is an axiom of theory T. Thus, in the third case, we have that

$$\mathsf{WSC}(CanMove(x); T; \{HasWheels, Car\}) \equiv HasWheels(x).$$

■

5.4 Generating Successor State Axioms

Example 5.4 Consider the problem of generating successor state axioms in a robot domain. This problem, in the propositional framework, is considered in [Lin, 2000]. On the other hand, a first-order formulation is much more natural and compact. We thus apply first-order logic rather than the propositional calculus and introduce the following relations, instead of propositions as considered in [Lin, 2000][6] :

- $Move(o, i, j)$ - the robot is performing the action of moving the object o from location i to location j
- $At(o, i)$ - initially, the object o is in the location i
- $At1(o, j)$ - after the action $Move(o, i, j)$, the object o is in location j
- $AtR(i)$ - initially, the robot is at location i
- $AtR1(j)$ - after the action $Move(o, i, j)$, the robot is at location j
- $H(o)$ - initially, the robot is holding the object o
- $H1(o)$ - after the action $Move(o, i, j)$, the robot is holding the object o.

Assume that the background theory contains the following axioms, abbreviated by T:

$$\forall o.(At(o, 1)) \wedge \forall o.(\neg At(o, 2)), \\ \forall o.[H(o) \equiv H1(o)], \\ \forall o, i, j.[(AtR(i) \wedge At(o, i) \wedge H(o) \wedge Move(o, i, j)) \rightarrow \\ (AtR1(j) \wedge At1(o, j))].$$

The goal is to find the weakest sufficient condition on the initial situation ensuring that the formula $At1(package, 2)$ holds. Thus we consider

$$\mathsf{WSC}(At1(package, 2); T; \{H, At, AtR, Move\}).$$

The approach we propose is based on the observation that

$$\mathsf{WSC}(At1(package, 2); T; \{H, At, AtR\}) \equiv \\ \forall H1 \forall At1 \forall AtR1.(\bigwedge T \rightarrow At1(package, 2))$$

After some simple calculations which can be performed automatically using the DLS algorithm we ascertain that $\mathsf{WSC}(At1(package, 2); T; \{H, At, AtR, Move\})$ is equivalent to:

$$[\forall o.At(o, 1) \wedge \forall o.\neg At(o, 2)] \rightarrow [H(package) \wedge \\ \exists i.(AtR(i) \wedge At(package, i) \wedge Move(package, i, 2))],$$

which, in the presence of axioms of theory T reduces to:

$$[H(package) \wedge \\ \exists i.(AtR(i) \wedge At(package, i) \wedge Move(package, i, 2))] \quad (20)$$

and, since $At(package, i)$ holds in the theory T only for i equal to 1, formula (20) reduces to:

$$H(package) \wedge AtR(1) \wedge Move(package, 1, 2).$$

Thus, the weakest condition on the initial state, making sure that after the execution of an action the package is in location 2, expresses the requirement that the robot is in location 1, holds the package and that it executes the action of moving the package from location 1 to location 2. ■

[6] Note that even this formalization can be generalized further for more than one action and transition, but we retain the correspondence to the original example for clarity.

6 Forgetting and Quantifier Elimination

Forgetting is considered in [Lin and Reiter, 1994; Lin, 2000] as an important technique for database progression and computing wsc's and snc's.[7] Given a theory T and a relation symbol P, forgetting about P in T results in a theory with a vocabulary not containing P, but entailing the same set of sentences that are irrelevant to P. Observe that forgetting is simply a second-order quantification, as shown in [Lin and Reiter, 1994]. Namely,

$$forget(\phi; P) = \exists P.\phi(P).$$

It is no surprise then that forgetting is not always reducible to first-order logic. On the other hand, due to Theorem 2.2, second-order quantifiers can often be eliminated resulting in a fixpoint or first-order formula.

Consider the following example.

Example 6.1 Let T consist of the following two axioms:

$$\forall x, y.[Mother(x, y) \rightarrow \exists z.Father(z, y)],$$
$$\forall x, y.[Father(x, y) \rightarrow Parent(x, y)].$$

Forgetting about $Father$ results in the second-order formula $\exists Father. \bigwedge T$ which, according to Theorem 2.2, is equivalent to:

$$\forall x, y.[Mother(x, y) \rightarrow \exists z.Parent(z, y)].$$

■

7 Conclusions

Using Lin's work as a starting point, we have provided new definitions for weakest sufficient and strongest necessary conditions in terms of 2nd-order formulas and provided the basis for algorithms which are guaranteed to automatically generate wsc's and snc's for both the propositional case and a non-trivial fragment of the first-order case. For the general propositional case, propositional snc's and wsc's are always generated using the techniques and for the semi-Horn fragment are always generated efficiently. For the first-order case restricted to the non-recursive semi-Horn fragment, reduction of wsc's and snc's to first-order formulas is always guaranteed and can be done efficiently. For the first-order case restricted to the recursive semi-Horn fragment, reduction to fixpoint formulas is always guaranteed and can be done efficiently. For the general first-order case, the techniques can also be applied, but reductions are not always guaranteed, even though the algorithm will always terminate.

This work generalizes that of Lin which only deals with the propositional case and it provides more direct methods for generating snc's and wsc's via syntactic manipulation. We have also demonstrated the potential use of this idea and these techniques by applying them to a number of interesting applications. Finally, we have re-interpreted the notion of forgetting in terms of quantifier elimination, shown how our techniques can be applied to a first-order version of forgetting and applied the technique to generation of successor-state axioms in restricted first-order situation calculus based action theories.

As a final observation, the quantifier elimination algorithm considered here has been implemented as an extension to the original DLS algorithm described in [Doherty *et al.*, 1997], for both the propositional and 1st-order cases.

[7] The forgetting operator for propositional logic is well-known in the literature as *eliminant* (see [Brown, 1990]).

References

[Abiteboul *et al.*, 1996] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley Pub. Co., 1996.

[Brown, 1990] F.M. Brown. *Boolean Reasoning*. Kluwer Academic Publishers, Dordrecht, 1990.

[Cadoli, 1995] M. Cadoli. *Tractable Reasoning in Artificial Intelligence*, volume 941 of *LNAI*. Springer-Verlag, Berlin Heidelberg, 1995.

[Dijkstra, 1976] E. W Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.

[Doherty *et al.*, 1996] P. Doherty, W. Lukaszewicz, and A. Szalas. A reduction result for circumscribed semi-Horn formulas. *Fundamenta Informaticae*, 28(3-4):261–271, 1996.

[Doherty *et al.*, 1997] P. Doherty, W. Lukaszewicz, and A. Szalas. Computing circumscription revisited. *Journal of Automated Reasoning*, 18(3):297–336, 1997.

[Doherty *et al.*, 1998] P. Doherty, W. Lukaszewicz, and A. Szalas. General domain circumscription and its effective reductions. *Fundamenta Informaticae*, 36(1):23–55, 1998.

[Ebbinghaus and Flum, 1995] H-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, Heidelberg, 1995.

[Kautz and Selman, 1996] H. Kautz and B. Selman. Knowledge compilation and theory approximation. *Journal of the ACM*, 43(2):193–224, 1996.

[Lin and Reiter, 1994] F. Lin and R. Reiter. Forget it! In R. Greiner and D. Subramanian, editors, *Working Notes of AAAI Fall Symposium on Relevance*, pages 154–159, Menlo Park, Ca., 1994. AAAI.

[Lin and Reiter, 1997] F. Lin and R. Reiter. How to progress a database. *Artificial Intelligence*, 92(1-2):131–167, 1997.

[Lin, 2000] F. Lin. On strongest necessary and weakest sufficient conditions. In A.G. Cohn, F. Giunchiglia, and B. Selman, editors, *KR2000*, pages 167–175, 2000.

[McCarthy, 2000] J. McCarthy. Approximate objects and approximate theories. In A.G. Cohn, F. Giunchiglia, and B. Selman, editors, *KR2000*, pages 519–526, 2000.

[Nonnengart and Szalas, 1998] A. Nonnengart and A. Szalas. A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. In E. Orlowska, editor, *Logic at Work: Essays Dedicated to the Memory of Helena Rasiowa*, volume 24 of *Studies in Fuzziness and Soft Computing*, pages 307–328. Springer Physica-Verlag, 1998.

[Nonnengart *et al.*, 1999] A. Nonnengart, H.J. Ohlbach, and A. Szalas. Elimination of predicate quantifiers. In H.J. Ohlbach and U. Reyle, editors, *Logic, Language and Reasoning. Essays in Honor of Dov Gabbay, Part I*, pages 159–181. Kluwer, 1999.