A Predicate-logic Version Method for Cost-based Hypothetical Reasoning Employing an Efficient Propositional-level Reasoning Mechanism

Mitsuru Ishizuka Kazuya Tanahashi Helmut Prendinger
Dept. of Information and Commun. Eng.
School of Engineering, University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan

Abstract

For propositional-level hypothetical reasoning (or abduction), particularly for its cost-based variant, some efficient methods have been developed which achieve polynomial-time reasoning (with respect to problem size) to compute a near-optimal solution. However, for predicate-logic version hypothetical reasoning, which allows rich and compact knowledge representation, it seems difficult to find a method with polynomial-order efficiency, as long as we stick to symbolic manipulation. As a result, there exists no efficient method in the predicate logic domain so far.

A naive approach for efficient predicate-logic version hypothetical reasoning might work as follows: knowledge expressed by predicate logic is first transformed into propositional knowledge, and then an efficient method is applied to the propositional knowledge base. If the knowledge base is function-free, this can be accomplished by instantiating variables to elements of the Herbrand universe (in this case simply a finite set of constants); however, this is impractical since a vast number of propositional clauses will be produced. We tried to use a deductive database technique, namely, the QSQR method, to extract the knowledge portion being related or relevant to the proof of a given goal of a hypothetical reasoning problem, and then to transform only this portion into propositional knowledge. Nevertheless, this approach cannot improve reasoning efficiency enough because QSQR requires considerable computational time for the transformation into propositional knowledge, and the number of the resulting propositional clauses is still large.

In this paper, we propose a *knowledge reformation* method for Horn rules expressed in predicate logic such that the transformation to propositional rules is efficient and the number of the resulting propositional rules is small. This knowledge reformation method is based on unfolding/folding procedures applied to predicate-logic Horn rules. Although its range of applicability is still limited, it is shown experimentally that, by introducing this type of knowledge reformation, a large amount of efficiency improvement is achieved for predicate-logic version cost-based hypothetical reasoning.

1 Introduction

While hypothetical reasoning (or abduction) is a useful framework for knowledge-based systems, its computational complexity is NP-complete or NP-hard [Kautz89, Bylander89]; consequently, its slow inference speed has been the most crucial problem in practice. However, regarding propositional-level hypothetical reasoning, some efficient inference mechanisms have been explored to date. In particular, for *cost-based hypothetical reasoning* (or abduction), efficient mechanisms have recently been developed. They can compute a near-optimal solution in polynomial time with respect to problem size [Ishizuka94, Ohsawa97, Ishizuka98, Santos94]. These methods take advantage of using search mechanisms borrowed from mathematical programming, a programming paradigm that may exploit the continuous rather than the discrete value domain. Besides hypothetical reasoning, several effective mechanisms have also been developed for other kinds of

propositional-level reasoning, e.g. SAT (Satisfiability Problem) or CSP (Constraint Satisfaction Problem), such as GSAT [Selman93a, 94b], Gu's method [Gu93], etc. In general, propositional-level reasoning is considered to be fundamental for many AI problems [Selman98].

On the other hand, in many cases the expressiveness of propositional logic as a knowledge representation language is not sufficient and a predicate-logic representation with variables is required. If possible at all, hundreds or thousands of propositional formulas are sometimes required to obtain a representation logically equivalent to a few predicate-logic formulas.

Although the expressiveness of predicate logic is high, it is hard to achieve efficient reasoning. For predicate-logic version hypothetical reasoning without function symbols but with inconsistency constraints, we have previously developed a reasoning system [Kondo93,96] which is based on efficient symbolic manipulation. Yet, it cannot overcome the wall of inference time growing exponentially. NBP (*Networked Bubble Propagation*) method, originally developed as a polynomial-time inference mechanism for cost-based propositional-level hypothetical reasoning, has been extended to deal with predicate-logic knowledge bases in [Ohsawa97]. However, because of its complex mechanism it is hard to scale up the system to deal with the size of problems encountered in practice.

Since there exist good methods in the propositional-level domain, it may be natural to consider a method that first transforms a predicate-logic knowledge base to an equivalent propositional logic knowledge base and then applies an efficient reasoning method to the propositional knowledge base. In principle, this can be accomplished by a transformation which instantiates variables in predicate-logic formulas to constants (ground terms), the elements of the Herbrand universe. (Note that function-free predicate-logic formulas can be converted into a finite number of propositional formulas.) This is, however, impractical since a vast number of propositional formulas will be produced this way.

Previously, we have tried to extract the knowledge portion being related to the proof of a given goal (of a hypothetical reasoning problem) by a deductive database technique, namely, the QSQR method, and then expand only this portion into propositional knowledge. Unfortunately, it cannot significantly improve reasoning efficiency because the QSQR method spends considerable computational time for the expansion into propositional knowledge, and the number of the resulting propositional clauses is still large.

In this paper, we propose a knowledge reformation method for predicate-logic rules where the expansion into propositional rules is efficient and the number of the resulting propositional rules is small. That is, as depicted in Fig. 1, our system first applies knowledge reformation to the original predicate-logic rules, then extracts the knowledge portion being related to proving a given goal, expands this portion into propositional knowledge, and finally applies an efficient propositional-level hypothetical reasoning method (our NBP or SL method) to obtain a near-optimal solution.

Although equivalence-preserving transformations of logic programs have been studied before (for example, by [Tamaki84]), there has been no research from the viewpoint of enhancing the efficiency of the expansion into propositional knowledge.

2 Background Methods

We first describe some background methods which form the basis of our system.

2.1 Cost-based Hypothetical Reasoning

The knowledge base for hypothetical reasoning consists of background knowledge Σ, which is always true, that is, true regardless of other knowledge about the domain, and a set H denoting hypothetical knowledge which may contradict with other knowledge (via inconsistency constraints contained in Σ). Let h be a subset of H, i.e., $h \subseteq H$. The basic procedure of hypothetical reasoning is as follows. When a goal G is given, the hypothetical reasoning system first tries to prove G from Σ alone. If it fails, it attempts to find a set h satisfying the following conditions:

1. $\Sigma \cup h \vdash G$ (G can be proved from Σ and h);

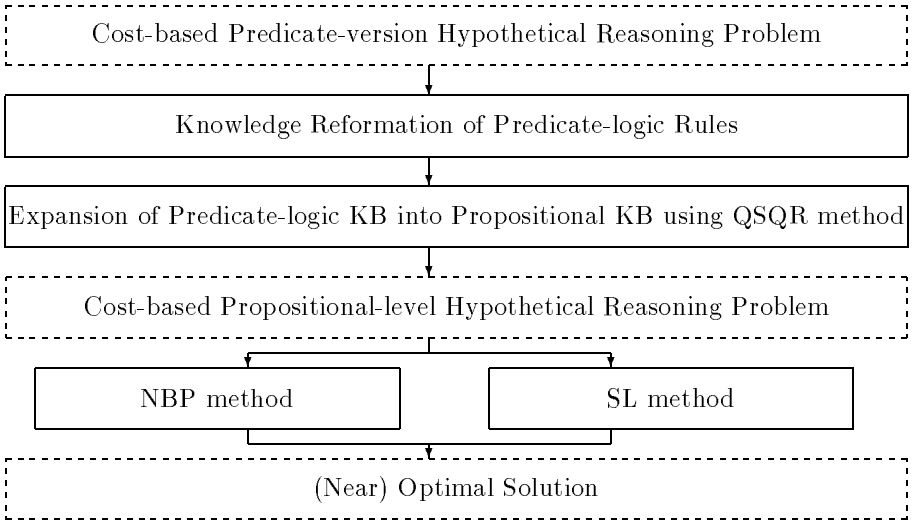

Figure 1: System configuration

2. $\Sigma \cup h$ is consistent.

When both conditions are satisfied, h is called a *solution hypotheses* set. In many cases, minimality (or non-redundancy) of h is required; that is, no proper subset h' of h ($h' \subset h$) should satisfy the above conditions. When a numerical weight is assigned to each element hypothesis and the cost of a solution is defined as the sum of the weights of the element hypotheses included in the solution hypothesis set, an optimal solution with minimal cost is to be searched by means of weighted or cost-based hypothetical reasoning [Charniak90, 94].

2.2 Networked Bubble Propagation (NBP) Method

As a polynomial-time method for computing a near-optimal solution for cost-based hypothetical reasoning, we first developed a method [Ishizuka94] employing the pivot-and-complement method [Balas80] from the area of 0-1 integer programming. The *NBP method* [Ohsawa97] utilizes the behavior of the pivot-and-complement method to guide the search behavior on a knowledge network. It achieves higher inference efficiency by exploiting structural features of a given problem as encoded in the knowledge network. It can achieve low-order polynomial-time inference of approximately $N^{1.5}$, where N is the number of propositional variables appearing in the problem formulation.

The NBP method is extended to deal with predicate-logic knowledge bases by considering the argument terms of predicates as network nodes. However, since the complexity of the network increases in the predicate-logic case, it is hard to scale up the system to deal with problems of practical size.

2.3 Slide-down & Lift-up (SL) Method

The pivot-and-complement method and the NBP method demonstrate the usefulness of first using the simplex method to determine an initial search point in a multi-dimensional continuous space, and then employing efficient local search mechanisms around this initial search point to find a near-optimal binary (1/0) solution. The local search mechanisms of the pivot-and-complement method and the NBP method are not, however, easy to understand.

In the *SL method* [Ishizuka98], a comprehensive local search mechanism is implemented which uses a non-linear programming technique and an effective local handler that enables the escape from traps into locally minimal points. This makes the local search mechanism simpler and comprehensive. The inference speed of SL method is approximately $N^{1.85}$, which is at present slightly slower than that of the NBP method. However, in terms of required memory space and scalability, due to its simple mechanism the SL method has some advantages over the NBP method.

2.4 QSQR Method

While the magic set method is well known as an efficient deductive database technique, the QSQR (Query and Subquery, Recursive) method [Vielle88] can achieve comparable performance, working in a goal-directed bottom-up fashion to compute all solutions. In this method, variable binding information synthesized in a bottom-up fashion in the search branch is propagated laterally to its brother nodes to narrow down search space, and synthesized subgoals are stored as lemmas which are used in the subsequent inference process to avoid redundant recalculation. In our approach, we use the QSQR method to extract the knowledge portion being related (relevant) to proving the goal of a given hypothetical reasoning problem (in a goal-directed bottom-up manner), and then to expand the predicate-logic knowledge base into a propositional knowledge base.

3 Transformation of a Predicate-logic Version Hypothetical Reasoning Problem into a Propositional One

In this section, we consider a transformation for solving predicate-logic version hypothetical reasoning problems by using an efficient propositional-level hypothetical reasoning method (i.e., the NBP or the SL method). In principle, function-free predicate-logic clauses can be expressed by a finite number of propositional clauses (by instantiating variables to constants). However, this is impractical because a vast number of propositional clauses will be produced.

Here, we use the QSQR method (explained in the previous section) in order to extract the predicate knowledge portion being possibly relevant to prove a given goal of a hypothetical reasoning problem, and to transform only this portion into propositional knowledge.

The cost-based hypothetical reasoning problems considered below consist of the following components:

- background knowledge consisting of function-free predicate-logic Horn clauses,
- constraints expressing inconsistent combinations of hypotheses, and
- element hypotheses which are ground-instantiated unit clauses with numerical weights (default weight=1).

(Rule-type hypotheses are sometimes required; however, they can alternatively be represented by means of rule-type background knowledge and unit-clause hypotheses. Both representations are equivalent in the sense that they imply the same set of non-hypothesis atoms. Thus, the above condition for representing hypothetical reasoning problems does not restrict the range of problems.)

A goal of a hypothetical reasoning problem is given as a ground-instantiated unit clause or a conjunction of them. The ground-instantiated portion of the knowledge base related to proving the given goal can be extracted through the process of the QSQR method; thereby the inconsistency constraints are ignored. The resultant portion of the knowledge base will consist of ground-instantiated unit-clause hypotheses and ground-instantiated predicate-logic Horn clauses. Then these ground-instantiated predicate-logic clauses are expanded into propositional clauses. The inconsistency constraints are transformed by plugging in ground-instantiated element hypotheses to element hypotheses containing variables, and then by replacing the resulting inconsistency constraints by their propositional counterparts. In this process, we can restrict the inconsistency constraints to those that contain element hypotheses shown to be relevant to the given goal. Thereby we can avoid an unnecessary increase in the number of inconsistency constraints.

By means of the above process, the relevant portion of a predicate-logic version hypothetical reasoning problem can be effectively expanded into a propositional hypothetical reasoning problem. To find a near-optimal solution for a hypothetical reasoning problem, we may now apply efficient propositional-level methods for cost-based hypothetical reasoning (the NBP or the SL method). Nevertheless, this approach will not necessarily achieve enough speed-up of the reasoning process, since considerable computational time is spent for the expansion into propositional clauses and the number of the resulting propositional rules (Horn clauses) is still large.

In order to cope with this problem, we introduce a *knowledge reformation* process that is to be applied to predicate-logic rule-type clauses of the original knowledge base. This reformation process aims at reducing the expansion time (of predicate-logic clauses into propositional clauses), and the number of the resulting propositional clauses. The process consists of three phases; i.e., (a) *unfolding phase*, (b) *1st folding phase* (association-variable separation phase), and (c) *2nd folding phase* (singleton-variable separation phase). Each of the three phases will be described below.

Note that knowledge reformation here is applied only to rule-type knowledge (the IDB predicates of a deductive database); the expansion of facts (unit clauses or the EDB predicates of a deductive database) and hypotheses is performed with the QSQR method.

First, we define different types of variables occurring in a rule. Consider the following rule in Prolog format.

$$\text{p(X,Z)} \text{ :- } \text{q1(X,Y), q2(Y,A), q3(Z).}$$

Head Variable a variable occurring as an argument of a head predicate. (X and Z in the above example.)

Non-head Variable a variable which is not a head variable.

Association Variable a non-head variable which occurs more than once as an argument in body predicates. (Y in the above example.)

Singleton Variable a non-head variable which occurs only once as an argument of a body predicate. (A in the above example.)

Substitution is defined as follows:

Substitution If there exist two rules A and B, and a body literal of A is unifiable with the head of B by unifier θ, then replace the body literal of A with the body of B and apply θ to the whole clause. If there are two or more body literals in A which are unifiable with the head of some clause, replace only the left-most one (the others are left unchanged).

3.1 Unfolding Phase

Consider the following knowledge base.

```
[I-i]  g(X,Z):- p(X,Y),q(Y,Z).
       q(Y,Z):- r(Y),s(Z).
[I-ii] g(X,Z):- p(X,Y),r(Y),s(Z).
```

The formula [I-ii] indicates the result of the unfolding operation applied to [I-i], where the body literal $q(Y,Z)$ of the 1st formula is substituted by the body of the 2nd formula. Suppose that there are three constants in the domain (a,b,c), and $p(X,Y)$ has 6 possible instantiations, for example, $p(a,a), p(a,b), p(a,c), p(b,a), p(b,b), p(b,c)$. Then we might compute the number of the generated propositional rules when the goal-directed expansion into propositional clauses is executed for the goal $g(X,Y)$. The number of produced propositional clauses can also be a measure to predict the time needed to perform the expansion into propositional knowledge; that is, the smaller the number is, the less time is needed for the expansion.

The 1st formula of [I-i] produces $6 \times 3 = 18$ propositional rules, and the 2nd formula of [I-i] produces $3 \times 3 = 9$ propositional rules. Thus, in this case, [I-i] will be transformed into $18 + 9 = 27$ propositional rules in total. On the other hand, from [I-ii], $6 \times 3 = 18$ propositional rules will be produced. If the goal is $g(a,b)$, the numbers of propositional rules to be produced are $3 + 3 = 6$ and 3 for [I-i] and [I-ii], respectively, since only the association variable Y is to be instantiated to a constant (either to a or b or c). As shown in the example, it is possible to reduce the number of the produced propositional rules with this kind of substitution or unfolding.

Concerning recursive rules such as

$$\text{g(X)} \text{ :- } \text{p(X,Y), g(Y).}$$

we do *not* unfold the recursive literal $g(Y)$, that is, we do not produce the formula

$$\text{g(X)} \text{ :- } \text{p(X,Y), p(Y,Y2), g(Y2).}$$

3.2 1st Folding Phase (Association-Variable Separation Phase)

The unfolding operation described above is not always effective. Consider a situation where the knowledge base contains a rule with two association variables.

```
[II-i]  g(X,Z) :- p1(X,Y1), p2(Y1,Y2), p3(Y2,Z).
[II-ii] g(X,Z) :- p1(X,Y1), q(Y1,Z).
        q(Y1,Z) :- p2(Y1,Y2), p3(Y2,Z).
```

Here, [II-i] can be obtained by replacing the body literal $q(Y1, Z)$ of the first formula in [II-ii] by the body of the second formula in [II-ii]; thus, [II-i] and [II-ii] are equivalent.

To keep the situation simple, we assume that each of the variables X, $Y1$, $Y2$ and Z allows for n ground instantiations such as $a, b, c, \cdots$. Transforming the predicate-logic rules into propositional rules in the goal-directed manner with respect to the goal $g(X, Z)$, we obtain n^4 rules in [II-i] and $2 \times n^3$ rules in [II-ii]. In this case, the number of propositional rules in [II-i] is $n/2$ times of the number of rules in [II-ii]; this means the form of [II-ii] becomes more advantageous over [II-i] as n increases.

As demonstrated in the previous example, if there are more than two association variables in one rule, we separate the rule into separate rules. Separation can be performed via a folding operation by means of introducing new predicate symbols. For example,

$$\text{g(X,Z)} \coloneq \text{p1(X,Y1), p2(Y1,Y2), p3(Y2,Z).}$$

is separated by introducing a new predicate-logic literal $new(X, Y)$ as follows,

$$\text{g(X,Z)} \coloneq \text{new(X,Y2), p3(Y2,Z).}$$
$$\text{new(X,Y2)} \coloneq \text{p1(X,Y1), p2(Y1,Y2).}$$

The separation operation of the association variables is executed as follows: first, the body part of the target rule is scanned from left to right; next, a new rule is introduced which contains the left-most separable association variable in its body literals; finally, the original clause is folded with the new rule. This operation is repeated until all separable association variables are processed. If there already exists a rule with a body that is identical (modulo variable renaming) to the body of the to be introduced (new) rule, this rule is not generated.

However, there are cases with association variables which are *not* separable, for instance, as in the rule

$$\text{g(X,Z)} \coloneq \text{p1(X,Y1,Y2), p2(Y1), p3(Y2,Z).}$$

In this case, association variable separation is not performed.

3.3 2nd Folding Phase (Singleton-Variable Separation Phase)

When a singleton variable with m possible instantiations exists in a rule, the number of generated propositional rules is m times larger. This can be avoided by reforming the rule including this singleton variable into separate rules.

For example,

$$\text{g(X,Y)} \coloneq \text{p1(X,Z), p2(Z,Y), p3(A).}$$
$$(A \text{ is a singleton variable.})$$

can be folded by introducing a new atom $new1$ to

$$\text{g(X,Y)} \coloneq \text{p1(X,Z), p2(Z,Y), new1.}$$
$$\text{new1} \coloneq \text{p3(A).}$$

In the case where k propositional rules are generated from the 1st predicate rule and m propositional rules from the 2nd rule, the total number of propositional rules is $k+m$, whereas $k \times m$ propositional rules will be generated from the original predicate-logic rule.

A rule containing a literal such as $p3(Z, A)$, where A is a singleton but Z is a non-singleton variable, cannot be separated; on the other hand, a clause containing a body literal such as $p3(c, A)$, where c is a constant and no variable besides the singleton variable occurs, can be folded into separate rules.

3.4 Application Order of the Three Types of Reformation

We apply the three reformation types described above in the following order:

1. unfolding phase
2. 1st folding phase (association-variable separation phase)
3. 2nd folding phase (singleton-variable separation phase)

At first unfolding operations are performed, resulting in rules with typically long rule bodies. Then those rule bodies are separated by possibly separating association variables and singleton variables. The effectiveness of knowledge reformation as described in this paper depends critically on the form of the knowledge base. We do not claim that this kind of knowledge reformation is effective for all knowledge base forms. Rather, its effectiveness is still limited; other effective reformation procedures are to be explored to cover a wider range of knowledge base forms. However, the knowledge reformation procedures discussed in this paper yield a good speed-up effect in many cases.

4 Effect of Knowledge Reformation

In this section, we illustrate how the knowledge reformation process described above reduces the total inference time in predicate-logic version hypothetical reasoning problems.

Figure 2 shows an example knowledge base. Here, if *double_loop*(a,b) is given as a goal, the system tries to find an optimal double loop such that (i) one loop starts from $X = a$ and goes back to $X = a$ via $Y1$, $Y2$, $Y3$, $Y4$ (*link*1, *link*2, *link*3 and *link*4), (ii) another loop starts from $Z = b$ and goes back to $Z = b$ via $Y1$, $Y2$, $Y3$, $Y4$ (*link*1, *link*2, *link*3 and *link*4), (iii) the used links do not violate the set of defined inconsistency constraints, and (iv) the sum of the weights of the used links is minimal. Figure 2 shows the case where each of $Y1, ..., Y4$ is instantiated to one of 5 values $(a0, ..., a4)$ and each link, such as, for instance, *link*1 between X and $Y1$ has 5×5 possibilities. We can change the problem size by changing the number of the possible values of $Y1 - Y4$ together with the number of their connecting links. (In the case of Fig. 2, the number of element hypotheses is 125; however, since each of the two loops selects this number of element hypotheses, the substantial number of element hypotheses is 250.)

The bottom part of Fig. 2 shows the reformed version of the original knowledge base shown at the top of the figure. In the knowledge reformation process, at first the following long rule is generated in the unfolding phase:

```
double_loop(X,Z) :-
  link1(X,Y1), link2(Y1,Y2), link3(Y2,Y3),
  link4(Y3,Y4), link5(Y4,X),
  link1(Z,YY1), link2(YY1,YY2), link3(YY2,YY3),
  link4(YY3,YY4), link5(YY4,Z).
```

Then, in the 1st folding phase, this rule is folded into several rules. Here, a separable association variable is separated one by one to generate a new rule. This separation operation is carried out from left to right. Observe that the body of the rule (generated in the unfolding phase) contains two identical subsequences $link1, ..., link5$ of literals (modulo variable renaming). As a consequence,

```
/***** Original Knowledge Base (before Reformation) *****/
double_loop(X,Z) :- loop(X),loop(Z).
loop(X) :- link1(X,Y1),link2(Y1,Y2),link3(Y2,Y3),
           link4(Y3,Y4),link5(Y4,X).
inc :- link1(X,Y),link2(Y,X).    /* constraint indicating inconsistency
inc :- link2(X,Y),link3(Y,X).    /* constraint indicating inconsistency
inc :- link3(X,Y),link4(Y,X).    /* constraint indicating inconsistency
inc :- link4(X,Y),link5(Y,X).    /* constraint indicating inconsistency
inc :- link5(X,Y),link1(Y,X).    /* constraint indicating inconsistency
/* Element hypotheses with associated weights (a value after /)
link1(a0,a0)/3.
link1(a0,a1)/9.
link1(a0,a2)/1.
link1(a0,a3)/5.
link1(a0,a4)/2.
link1(a1,a0)/10.
link1(a1,a1)/3.
link1(a1,a2)/5.
link1(a1,a3)/10.
link1(a1,a4)/10.
link1(a2,a0)/9.
link1(a2,a1)/9.
link1(a2,a2)/1.
link1(a2,a3)/5.
link1(a2,a4)/10.
link1(a3,a0)/5.
link1(a3,a1)/4.
link1(a3,a2)/3.
link1(a3,a3)/7.
link1(a3,a4)/3.
link1(a4,a0)/2.
link1(a4,a1)/10.
link1(a4,a2)/5.
link1(a4,a3)/2.
link1(a4,a4)/4.
/* Hypotheses regarding link2, link 3, link4 and link5
/* are similar to link1, and are omitted here.

/***** Knowledge Base after Reformation *****/
double_loop(X,Z) :- nn4(X),nn3(Z,XX6),link5(XX6,Z).
nn4(X) :- nn3(X,Y4),link5(Y4,X).
nn3(X,Y4) :- nn2(X,Y3),link4(Y3,Y4).
nn2(X,Y3) :- nn1(X,Y2),link3(Y2,Y3).
nn1(X,Y2) :- link1(X,Y1),link2(Y1,Y2).
/* No change of inconsistency constraints and element hypotheses,
/* which remain the same as in the original knowledge base.
```

Figure 2: An example knowledge base (before and after reformation).

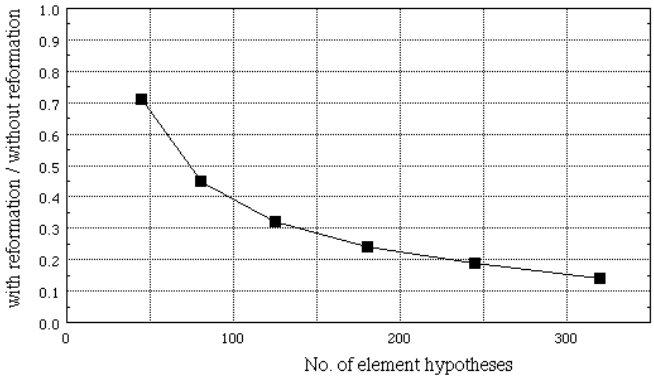

Figure 3: Reduction effect of the number of expanded propositional rules with knowledge reformation (in the case of the knowledge base of Fig.2).

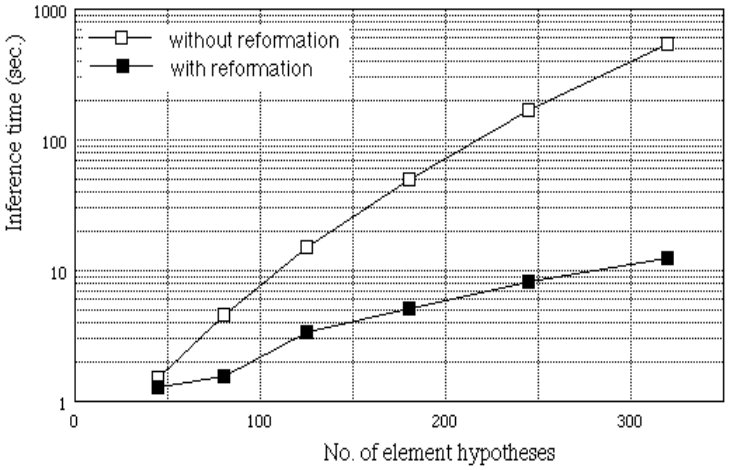

Figure 4: Efficiency improvement of total inference time with knowledge reformation (in the case of the knowledge base of Fig.2).

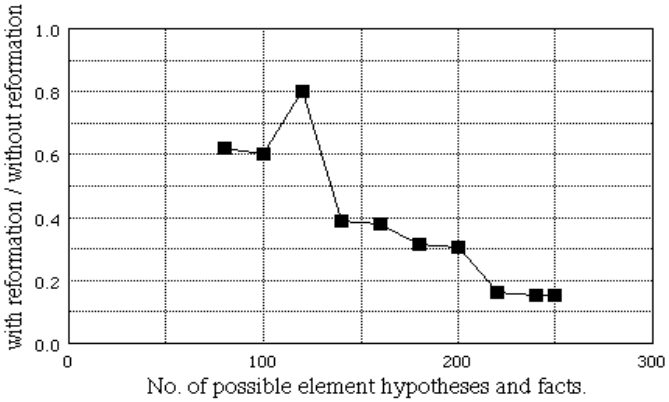

Figure 5: Reduction effect of the number of expanded propositional rules with knowledge reformation (in another knowledge-base case).

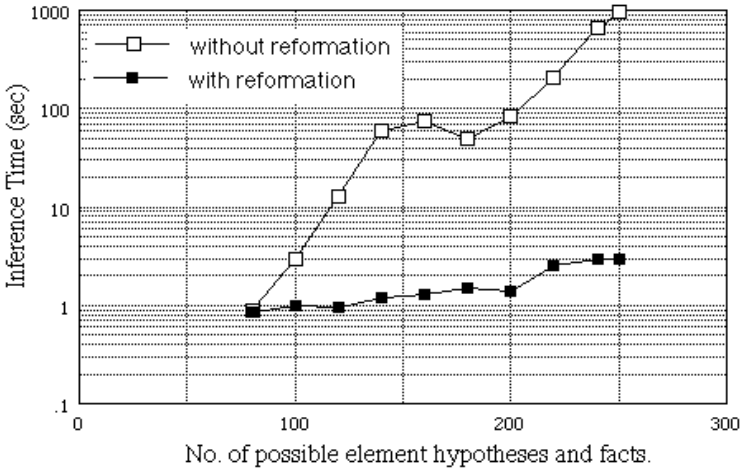

Figure 6: Efficiency improvement of total inference time with knowledge reformation (in another knowledge-base case).

separation of association variables regarding the second subsequence does not produce additional rules since the corresponding new rules have already been produced for the first subsequence.

Figure 3 depicts the ratio of the numbers of the generated propositional rules from the example predicate-logic knowledge base with and without knowledge reformation. As the problem scales up, the reduction rate of the number of propositional rules expanded increases with knowledge reformation.

Figure 4 depicts the experimental result of the total inference time spent for the expansion into propositional knowledge, and the computation of a near optimal solution using NBP method, a polynomial-time reasoning method for cost-based propositional hypothetical reasoning. The time used for knowledge reformation is included in the total time for the 'with reformation' case, but it is very small. Figure 4 shows that the total inference time reducing effect of knowledge reformation performed on the predicate-logic rules is significant, and the effect becomes stronger as the problem scales up.

About 90% of the total inference time is spent for expanding the predicate-logic knowledge base into a propositional knowledge base by using the QSQR method. Assuming that the number of propositional variables (N) is proportional to the number of propositional rules, since the computation time of NBP method is $O(N^{1.5})$, the inference time of NBP method reduces to $1/n^{1.5}$ as the number of propositional rules becomes $1/n$.

The inference time of KICK-HOPE II [Kondo96] is slightly longer (2 or 3 times) than that of the 'without reformation' case. KICK-HOPE II is an efficient predicate-logic hypothetical reasoning system that computes an optimal solution hypotheses set employing predicate-level symbolic manipulation.

Figure 4 illustrates the advantage of the proposed method quantitatively, but only with respect to a particular example. However, the proposed method brings about a similar effect for many other problems. For example, Figures 5 and 6 show experimental results for a predicate-logic knowledge base where various types of rules are included. While some fluctuation exists, the reduction of inference time becomes very significant when the size of the problem increases.

5 Conclusion

Although some efficient methods have been explored for propositional-level inference problems, it is still hard to find an efficient method for problems that are formulated in a more expressive language such as predicate logic. In this paper, we have presented a new efficient method for cost-based predicate-logic hypothetical reasoning, where a predicate-logic knowledge base is first expanded into a propositional knowledge base and then an efficient propositional-level method is applied to compute a near-optimal solution. In particular, we have shown that knowledge reformation of predicate-logic rules is very effective as a preprocessing phase in order to reduce the number of generated propositional rules, and consequently to reduce the total inference time.

Issues remaining for future research are as follows.

- The range of applicability of the three types of the knowledge reformation as described in this paper is still limited. Moreover, the quantitative evaluation of their effectiveness is not completely clear. We have to exploit more powerful kinds of knowledge reformation that are effective for a wider range of knowledge base forms, as well as clarify their performance quantitatively.

- We have used the QSQR method to extract the knowledge portion being related or relevant to the proof of a given goal, and to expand the extracted predicate-logic knowledge base into a propositional one. The QSQR method is a deductive database technique and not completely suitable for this purpose. We have to develop a more efficient method to carry out these processes.

References

[Balas80] E.Balas and C.Martin: Pivot and Complement—a Heuristic for 0-1 Programming, *Management Science*, Vol.26, pp.86–96 (1980)

[Bylander89] T.Bylander et al.: Some Results Concerning the Complexity of Abduction, *Proc. 1st Conf. on Principles of Knowledge Representation and Reasoning (KR'89)* (1989)

[Charniak90] E.Charniak and S.Shimony: Probabilistic Semantics for Cost Based Abduction, *Proc. AAAI-90*, pp.106–111 (1990)

[Charniak94] E.Charniak and S.Shimony: Cost-Based Abduction and MAP Explanation, *Artificial Intelligence*, Vol.66, No.2, pp.345–374 (1994)

[Gu93] J.Gu: Local Search for Satisfiability(SAT) Problem, *IEEE Trans. SMC*, Vol.23, No.4, pp.1108–1129 (1993)

[Ishizuka94] M.Ishizuka and T.Okamoto: A Polynomial-time Hypothetical Reasoning employing an Approximate Solution Method of 0-1 Integer Programming for Computing Near-optimal Solution, *Proc. 10th Canadian Conf. on AI*, Banff, pp.179–186 (1994)

[Ishizuka98] M.Ishizuka and Y.Matsuo: SL Method for Computing a Near-optimal Solution using Linear and Non-linear Programming in Cost-based Hypothetical Reasoning, *PRICAI'98: Topics in Artificial Intelligence (Proc. PRICAI'98, Singapore) (H.Y.Lee and H.Motoda (eds.))*, pp.611-625, Springer, (Lecture Notes in AI 1531) (1998)

[Kautz89] H.Kautz and B.Selman: Hard Problems for Simple Default Logics, *Proc. 1st Conf. on Principles of Knowledge Representation and Reasoning (KR'89)* (1989)

[Kondo93] A.Kondo, T.Makino and M.Ishizuka: Efficient Hypothetical Reasoning System for Predicate-logic Knowledge Base, *Knowledge-Based Systems*, Vol.6, No.2, pp.87–94 (1993)

[Kondo96] A.Kondo and M.Ishizuka: Efficient Inference Method for Computing an Optimal Solution in Predicate-Logic Hypothetical Reasoning, *Knowledge-Based Systems*, Vol.9, pp.163–171 (1996)

[Ohsawa97] Y.Ohsawa and M.Ishizuka: Network Bubble Propagation: A Polynomial-time Hypothetical Reasoning Method for Comuting Near-optmal Solutions, *Artificial Intelligence*, Vol.91, No.1, pp.131–154 (1997)

[Santos94] E.Santos, Jr.: A Linear Constraint Satisfaction Approach to Cost-based Abduction, *Artificial Intelligence*, Vol.65, No.1, pp.1–28 (1994)

[Selman93a] B.Selman and H.Kautz: An Empirical Study of Greedy Local Search for Satisfiability Testing, *Proc. AAAI-93*, pp.46–51 (1993)

[Selman93b] B.Selman and H.Kautz: Domain-independent Extension to GSAT: Solving Large Structured Satisfiability Problems, *Proc. IJCAI-93*, pp.290–295 (1993)

[Selman97] B.Selman, H.Kautz and D.McAllester: Ten Challenges in Propositional Reasoning and Search, *Proc. IJCAI-97*, pp.50–54 (1997)

[Tamaki84] H.Tamaki and T.Sato: Unfold/fold Transformation of Logic Programs, *Proc. 2nd Int'l Logic Programming Conf.*, Uppsala, pp.127–138 (1984)

[Vielle86] L.Vieille: Recursive Axioms in Deductive Databases: The Query/Subsequery Approach, *Proc. 1st Int'l Conf. on Expert Database Systems*, pp.179–193 (1986)