Speedup of hypothetical reasoning by experience-based learning mechanism

Toshiro Makino* and Mitsuru Ishizuka

While hypothetical reasoning is a useful knowledge system framework that is applicable to many practical problems, its crucial problem is its slow inference speed. The paper presents a speedup method for hypothetical reasoning systems that uses an experience-based learning mechanism which can learn knowledge from inference experiences to improve the inference speed in subsequent inference processes, sharing subgoals similar to those of the prior inference processes.

This learning mechanism has common functions with existing explanation-based learning. However, unlike explanation-based learning, the learning mechanism described in the paper has a learning capability even at intermediate subgoals that appear in the inference process. Therefore, the learned knowledge is useful even in the case in which a new goal given to the system shares a subgoal that is similar to those learned in the prior inference. Since the amount of the learned knowledge becomes very large, the learning mechanism also has a function of learning selectively only at subgoals which substantially contribute to the speedup of the inference. In addition, the mechanism also includes a knowledge reformation function, which transforms learned knowledge into an efficient form to be used in subsequent inference.

Keywords: hypothetical reasoning, deductive learning, efficient inference

Incomplete knowledge plays an important role in expanding the capability of knowledge bases. Incomplete knowledge here means knowledge with exceptions, knowledge with the possibility of inconsistency, hypothetical knowledge, defeasible knowledge etc. A logic-based hypothetical reasoning system [1,2], which can handle incomplete knowledge as hypotheses, is an important framework for advanced knowledge systems, because of its theoretical basis and its practical usefulness. It is applicable to many problems, including diagnosis [1,3] and design [4]. However, one crucial problem with hypothetical reasoning is its slow inference speed.

In order to improve its inference speed, some methods have been proposed. They are, for example, assumption-based truth maintenance systems (ATMS) [5], a fast hypothetical reasoning method using an inference-path network [6], and a fast hypothetical reasoning method [7] applicable to predicate Horn clauses utilizing a deductive database technique. A key technique in these methods is the avoidance of inefficient backtracking or recalculation caused by inconsistency among adopted hypotheses.

However, since the computational complexity of non-monotonic reasoning including hypothetical reasoning has been proved to be NP-complete or NP-hard [8], we cannot overcome the limit of inference time increasing exponentially with respect to problem scales as long as we use ordinary inference methods. A method using an analogical inference [9] has been presented as one method of overcoming this inference speed limit.

This paper presents a speedup method using an experience-based learning mechanism for a hypothetical reasoning system with predicate Horn clauses and consistency constraints. This method allows the acquisition of effective knowledge from inference experience for speeding up subsequent inference that has some similarity to prior inferences. Unlike existing explanation-based learning (EBL) [10,11], the system based on this method can learn knowledge even at intermediate subgoals

Department of Information and Communication Engineering, Faculty of Engineering, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113, Japan
*Currently with NTT Communication Science Laboratories, Japan
Paper received 12 August 1992. Revised paper received 23 July 1993. Accepted 20 September 1993

appearing in the inference process. This function enables us to expand the effectiveness of experience-based or explanation-based learning. Since the amount of knowledge thus learned from inference experience becomes very large, the system also includes a function of selectively learning only at effective subgoals from the viewpoint of improving the inference speed, and a function of reforming learned knowledge into an efficient form for subsequent inference.

LOGIC-BASED HYPOTHETICAL REASONING AND EXPERIENCE-BASED LEARNING

Our hypothetical reasoning in this paper is a logic-based one first proposed by Poole *et al.* [1,2]. Knowledge in this system is divided into two categories, i.e. background knowledge denoted by Σ which is always true, and hypothesis knowledge denoted by H which is not always true, or defeasible knowledge. The hypothesis knowledge set H may include inconsistency, whereas the background knowledge set Σ has no possibility of inconsistency. The basic behavior of this system is to find a consistent subset h of H necessary to prove or explain a given goal G together with Σ. This can be written as $h \subseteq H$ (where h is a subset of H), $\Sigma \cup h \vdash G$ (where G can be proved from $\Sigma \cup h$), and $\Sigma \cup h \nvdash \Box$ (where $\Sigma \cup h$ is consistent). The h becomes a solution hypothesis set to satisfy the goal G. Here, the deductive inference mechanism is used in reverse direction to abductively generate a solution hypothesis h.

In general, knowledge here can be expressed with first-order predicate logic formulae. However, in order to achieve efficient inference, we restrict the knowledge expression to predicate Horn clauses and consistency constraints. The consistency constraints are introduced to express inconsistent combinations of hypotheses explicitly, since we cannot express logical negation explicitly, and therefore inconsistency with Horn clauses in the knowledge base.

The hypothetical reasoning system can be easily implemented by utilizing a backward (top-down) inference mechanism embedded in Prolog. In this case, a necessary set of hypotheses is generated along the depth-first inference path. This generated hypothesis set is then subjected to a consistency check. If an inconsistency is found, a part of the generated hypothesis set is discarded and another hypothesis is generated in accordance with the backtracking mechanism of Prolog. When the inference succeeds in proving a given goal, the generated consistent hypothesis set becomes a solution hypothesis set for the goal. Another type of inference method, i.e. the consequence-finding-type inference method, is described in Reference 12 for hypothetical or abductive reasoning. These simple implementations are, however, not necessarily efficient, particularly because of the backtracking caused by the inconsistency among adopted hypotheses. Thus, in order to improve efficiency, a notion of a goal-directed parallel forward (bottom-up) inference mechanism without backtracking is incorporated, for example, in References 6 and 7. In this paper, we show another way to speedup hypothetical reasoning, i.e. a method of the reformation of the knowledge base into an efficient one on the basis of the experience of inference processes.

An inference process for proving a given goal by adopting necessary hypotheses can be regarded as experience in hypothetical reasoning. Similar subgoals appear in different inference processes with respect to different goals. The similarity among the subgoals indicates here a relation in which two subgoals share the same generalized literal obtainable by changing the constant terms of literals into variables. Thus, if we record an inference result in one inference experience and store its generalized version as knowledge, we can utilize its knowledge to improve the inference speed in subsequent inference processes including subgoals similar to prior ones. We can acquire more widely useful knowledge by learning at subgoals rather than learning only at a final goal as in existing explanation-based learning. Since consistency maintenance among adopted hypotheses is required in the process of hypothetical reasoning, we have to take care of this function in the design of our experience-based learning mechanism.

In this paper, we use ‘experience-based learning’ for our method, to distinguish it from existing explanation-based learning (EBL).

STRUCTURE OF HYPOTHETICAL REASONING SYSTEM WITH EXPERIENCE-BASED LEARNING MECHANISM

We first describe the structure of a constructed hypothetical reasoning system with an experience-based learning mechanism. Basically, this system searches a single solution hypothesis set for a given goal in a backward depth-first manner. The mechanism of the experience-based learning will be described in the next three sections. *Figure 1* depicts the structure of the system, which is implemented as a meta-interpreter of Sicstus-Prolog. We will briefly describe the function of each component.

- *Inference engine:* This inference engine tries to prove a given goal using background knowledge and hypotheses generated by a hypothesis generator. The proof strategy of this inference engine is backward and depth-first as in Prolog. When the goal proof is succeeded, it produces a consistent hypothesis set adopted in the inference process as a solution hypothesis for the goal. Also, this engine generates a generalized goal proof tree based on a generalized

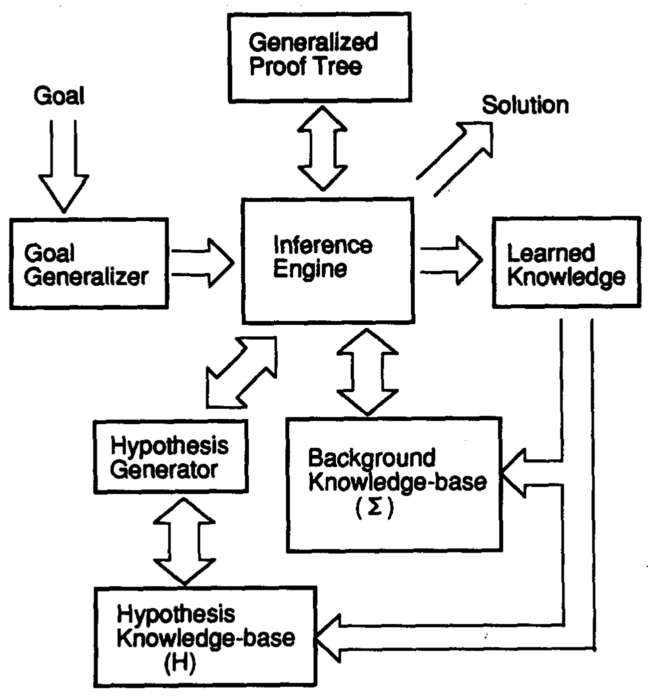

Figure 1 Hypothetical reasoning system with experience-based learning mechanism

goal provided by a goal generalizer. Using this generalized proof tree, it produces generalized knowledge, which is stored in the knowledge base.

- *Background knowledge base:* This knowledge base stores background knowledge (Σ) with no possibility of inconsistency with respect to other knowledge. Consistency checking is not required for this background knowledge. Each piece of background knowledge is expressed in the form of *bk*(predicate Horn clause). As a part of background knowledge, consistency constraints are written with predicate Horn clauses having an 'inc (inconsistent)' atom as their heads, for example, as *bk*(inc :- *p*(*a*), *q*(*b*).), which means that *p*(*a*) and *q*(*b*) cannot coexist in one environment.
- *Hypothesis knowledge base:* Hypothesis knowledge (*H*), which is not always true and possibly becomes inconsistent against other knowledge in certain situations, is stored separately in this hypothesis knowledge base. Necessary hypotheses in the process of hypothetical reasoning are generated from this knowledge base. Each possible hypothesis knowledge is expressed in the form of *hk*(knowledge-ID with variables, predicate Horn clause), where the variables attached to the knowledge ID correspond to those appearing in the predicate Horn clauses.
- *Hypothesis generator:* This generator generates the necessary hypothesis for the inference engine by selecting appropriate hypothesis knowledge and instantiating it if necessary.
- *Goal generalizer:* This goal generalizer produces a generalized goal by changing instances into variables. The generalized goal is given to the inference engine to learn generalized knowledge from the experience.
- *Generalized proof tree:* This is a generalized proof tree obtained, in principle, as a proof tree for the generalized goal. The structure of this generalized proof tree is the same as one obtained for the corresponding specific goal. Since the constructions of these two proof trees for the specific and generalized goals or subgoals are made simultaneously in the actual inference process, the computational cost of constructing the generalized proof tree is small.

EXPERIENCE-BASED LEARNING MECHANISM

Our hypothetical reasoning system has been enhanced with the following experience-based learning mechanism.

Learning from successful proof

In this case, knowledge is acquired basically as in explanation-based learning [10,11]. That is, the system first constructs a proof tree starting from a given specific goal. At the same time, a generalized proof tree having the same structure as the above tree is obtained for the corresponding generalized goal, which is generated by changing the constants instantiated for the specific goal to original variables appearing in original knowledge.

The proof tree for the specific goal here serves as a useful bias for producing the generalized knowledge, i.e. a generalized short-cut proof path. Then the generalized goal and a set of fact-type knowledge 'fact_*k*' (k = 1,2, . . .) necessary to satisfy this goal are identified from the generalized proof tree. In our experience-based learning mechanism, fact-type knowledge (unit clauses) in the knowledge base has an operationality that is a key concept in explanation-based learning.

Since hypotheses are generated in the proof process of hypothetical reasoning, knowledge thus learned is stored in the following two ways:

- When the goal is proved without adopting any hypotheses, knowledge acquired from the experience is stored in the background knowledge base (Σ) as *bk*(goal :- fact_1, fact_2, . . . , fact_*n*.).
- When more than one hypothesis is adopted in the proof process for the goal, acquired knowledge is stored together with the hypotheses in the hypothesis knowledge base *H* as *hk*([hyp_ID_1, . . ., hyp_ID_m], goal :- fact_1, . . ., fact_*n*.), where the 'hyp_ID_*k*' (k = 1, 2, . . .) denote hypothesis names with variables as their arguments, and 'fact_*k*' (k = 1,

2, . . . ,n) are fact-type pieces of knowledge necessary to satisfy the goal.

That is, in the first case, there is no possibility of inconsistency between the learned knowledge and other knowledge, since no hypotheses are involved. Thus it can be stored in the background knowledge base Σ. In contrast, in the second case, it is not certain that the adopted hypotheses are consistent with other knowledge in different conditions. Accordingly, the learned knowledge is stored with the list of accompanying hypotheses in the hypothesis knowledge base H. When this knowledge is used in subsequent inference processes, this list of hypotheses is evoked to check the consistency of knowledge.

The following correspondence exists between existing explanation-based learning and our experience-based learning described in this paper.

goal concept	$\leftrightarrow$ subgoals appearing in inference process
domain theory	$\leftrightarrow$ knowledge in background and hypothesis knowledge bases
training example	$\leftrightarrow$ specific goal given to hypothetical reasoning system
operationality	$\leftrightarrow$ fact-type knowledge

Basic differences from existing explanation-based learning are as follows.

- Knowledge learning takes place even at subgoals, and not only at the final goal, so that the system can acquire more widely useful knowledge from experience.
- Since the hypothetical reasoning process generates necessary hypotheses and executes consistency checking, knowledge learned from experience keeps a record of hypotheses that are necessary to validate the learned knowledge.

Because of the occurrence of inconsistency among adopted hypotheses, the number of backtrackings in hypothetical reasoning is larger than that for other reasoning schemes. Furthermore, consistency checking is computationally expensive. The use of the record of successful proofs as learned knowledge accompanied by necessary hypotheses contributes to a great extent to the reduction of backtracking and consistency checking in subsequent inference.

Learning from proof failure

There are two cases in which a proof fails. One is the case in which the system cannot find a matching predicate symbol in the heads of Horn clauses; another is the case in which necessary unification fails.

In the former case, the system records failed goals or subgoals after their generalization, since the proofs of these goals or subgoals always fail, not depending on the instantiation of their variables. In the latter case, failed goals or subgoals are recorded without generalization, since the failure is dependent on a specific instantiation of their variables. That is, learned knowledge from proof failure is stored in the knowledge base in the forms of

$$bk(\text{goal}(V_1, V_2, \ldots)\text{:- fail.}) \quad \{V_1, V_2, \ldots \text{ are variables}\}$$

for the former case, and

$$bk(\text{goal}(c_1, c_2, \ldots)\text{:- fail.}) \quad \{c_1, c_2, \ldots \text{ are constants}\}$$

for the latter case.

Most proof failures in practice fall into the latter category, since rules having a subgoal without a matching predicate symbol in the former case are useless ones that must be removed from the knowledge base. Therefore, we must mainly consider the latter case in practice.

SELECTIVE LEARNING AT SUBGOALS

Our learning mechanism can learn generalized knowledge at subgoals, and not only at a final goal, to improve the efficiency of subsequent inference processes. However, since the amount of this learned knowledge becomes too large, it is necessary to restrict the target subgoals at which the knowledge learning is evoked, according to a certain criterion. One way is to ask knowledge-base builders to define such target subgoals of learning. This may be a practical way, since most knowledge-base builders know the meaning and levels of knowledge (predicates), and may be able to specify key subgoals (predicates). The key subgoals (predicates) thus specified, however, are not necessarily effective ones from the viewpoint of speeding up hypothetical reasoning. Thus we will consider a systematic method of selecting effective target subgoals for our experience-based learning.

Basic strategy

Our hypothetical reasoning system in this paper tries to find a single solution hypothesis set in backward depth-first search mode, as in Prolog. The most serious inefficiency in this hypothetical reasoning comes from the backtracking caused by inconsistency among adopted hypotheses. Accordingly, it becomes effective to avoid this type of backtracking by learning knowledge at root subgoals of this backtracking. We will identify such subgoals for selective experience-based learning.

Background Knowledge(Σ)	Hypothesis Knowledge(H)
a :- b, d.	g.
b :- h.	h.
b :- i.	i.
c :- e.	j.
c :- f.	k.
d :- f, g.	l.
e :- i, j.	
f :- k, l.	
inc :- i, k.	

Figure 2 Knowledge base

As described above, knowledge denoting inconsistency constraints is written in our knowledge base as Horn clauses having 'inc' atoms as their heads. The above backtracking may occur at subgoals where one of these inconsistency constraints is applied to possibly remove an inconsistent combination of adopted hypotheses. These subgoals can be identified by analyzing the knowledge structure described in the knowledge base.

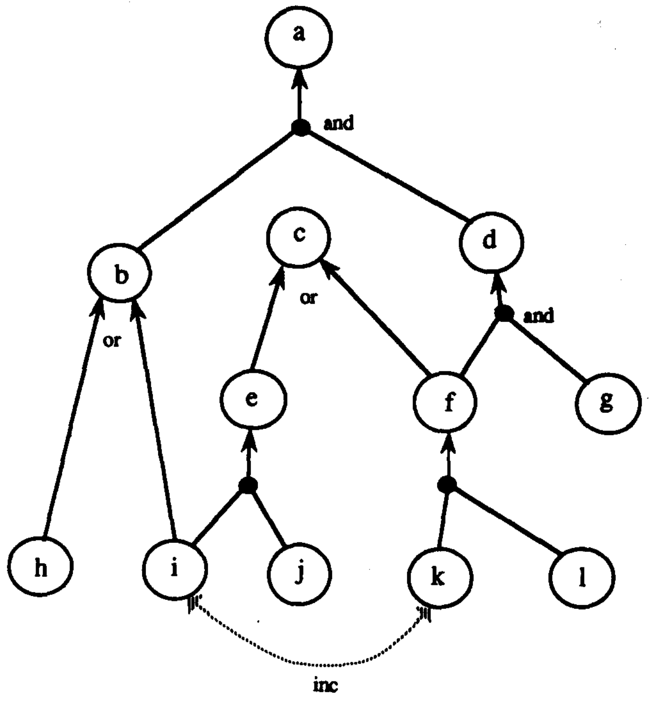

Figure 3 Networked knowledge expression for *Figure 2*

Propositional Horn clause case

We first consider a knowledge base expressed in propositional Horn clauses for simplicity, before considering a somewhat complicated case of predicate Horn clauses. In this case of propositional Horn clauses, the knowledge structure representing the relations of the described knowledge can be expressed as a network. For example, the knowledge set of *Figure 2* can be expressed as the network of *Figure 3*, in which an inconsistency constraint 'inc :- *i*, *k*.' is expressed by a link with 'inc' between nodes *i* and *k*.

The target subgoals for our experience-based learning are nodes where all the upward propagations along the network from every tip node of one 'inc' link meet with an AND relation. In *Figure 3*, node *a* is such a subgoal, whereas node *c* is not, since two upward propagations from nodes *i* and *k* meet with an OR relation, and not with an AND relation, at node *c*. This type of node or subgoal can be found by symbol manipulation as follows.

Let us construct lists of upward ancestor atoms starting from atoms that appear in the body part of one inconsistency constraint. For example, in the knowledge base of *Figure 2* where *i* and *k* appear in an inconsistency constraint 'inc :- *i*, *k*', the list of ancestor atoms for *i* is *b*, *e*, *a*, *c*, and the list of ancestor atoms for *k* is *f*, *c*, *d*, *a*. We detect common atoms in these lists. In this example, atoms *a* and *c* are these common atoms, which correspond to the nodes where the upward propagations from an 'inc' link meet. Then we check the body of Horn clause knowledge having these atoms as heads, that is, the following Horn clauses are checked:

a :- *b*, *d*.
c :- *e*.
c :- *f*.

Since the body of '*a* :- *b*, *d*.' include two atoms appearing in the lists of ancestor atoms for *i* and *k*, atom *a* is identified to be a target subgoal or node where the propagations from the 'inc' link meet with an AND relation. On the other hand, the body of '*c* :- *e*.' or '*c* :- *f*.' includes only an atom (atoms in general) appearing in the list of ancestor atoms for *i* or *k*. The propagations from the 'inc' link meet at *c* with an OR relation. Consequently, atom *c* is not considered to be a target subgoal for our learning.

Although we have described the algorithm to identify effective subgoals for learning in the propositional Horn clause case, we have to take into account other conditions for the predicate Horn clause case.

Predicate Horn clause case

Predicate Horn clauses including variable expressions are usually required to represent knowledge in many practical problems. Let us consider here function-free predicate Horn clauses. In this case, it is difficult to construct a knowledge network as in the case of propositional Horn clauses, because of the existence of variables and recursive loops. However, since what we need here is to determine target subgoals for our experience-based learning, we do not necessarily need to construct the knowledge network. We can determine them only by considering ancestor literals of hypothesis knowledge appearing in the body of inconsistency constraints.

Recursive loops

We can detect recursive loops when we generate a list of ancestor literals. That is, if a new ancestor literal found by tracing the connection of knowledge is the same as one found previously, then we can detect the existence of a loop. By avoiding the recursive part of the loop, we can generate finite ancestor literals for each piece of hypothesis knowledge related to inconsistency constraints.

Inconsistency constraints

The following deals with inconsistency constraints of the following type:

'inc;- $p(X), p(Y), X \backslash= Y$.'
{$\backslash=$ not equal, X and Y variables}

Because of the existence of arguments, it is conceivable that inconsistency occurs even between the same predicate literals as exemplified above. The lists of ancestor literals for $p(X)$ and $p(Y)$ are the same. Therefore, we cannot determine the target subgoals for learning, i.e. the subgoals where adopted hypothesis sets are possibly removed by inconsistency checking, simply by finding common predicate literals in the lists. However, if the body of a piece of Horn-clause knowledge whose head is a found ancestor literal includes plural elements (more than two elements in this case) of the list of ancestor literals, then we can determine that its head literal is a target subgoal, where the propagations from an 'inc' link meeting with an AND relation.

Generalization of learned knowledge depending on types of inconsistency constraint

Since learned knowledge is acquired with generalization from successful proofs in our experience-based learning, there are possibilities that the learned generalized knowledge yields inconsistency. Therefore, when the system uses this learned knowledge in the inference process, there remains the possibility of backtracking. The backtrackings that occur at the portion of learned knowledge are shallow in general; they are not as serious as in the case of original knowledge. It is, however, better to avoid these backtrackings. Thus we determine whether or not to execute the generalization of knowledge acquired at a certain subgoal, depending on the types of inconsistency constraint affecting the consistency checking at the subgoal, as follows:

- 'inc :- $p(X)$, $q(Y)$.': For this type of inconsistency constraint, inconsistency always occurs when $p(X)$ and $q(Y)$ coexist in one adopted hypothesis set, not depending on the instantiations of variables X and Y. Therefore, generalized knowledge acquired from a successful proof never invokes backtracking due to the above type of inconsistency constraint. Thus, we generalize knowledge acquired at a subgoal determined as a learning target by this type of inconsistency constraint.
- 'inc :- $p(X)$, $q(Y)$, $X \backslash = Y$.': In this case, inconsistency occurs depending on the instantiation of X and Y. Therefore, generalized knowledge acquired from a successful proof may possibly invoke backtracking in some different instantiations of the variables. Accordingly, we do not generalize knowledge acquired at a subgoal determined by this type of inconsistency constraint.

REFORMATION OF LEARNED KNOWLEDGE

The amount of knowledge increases by the accumulation of learned knowledge from experiences. In particular, a large number of pieces of rule-type knowledge having the same head predicate symbol may be generated by learning. As a result, the efficiency of the inference may sometimes decline, because the system has to repeat backtrackings until it finds an appropriate piece of knowledge. In order to reduce this sort of inefficiency in our hypothetical reasoning system, we introduce a reformation function of learned knowledge into our system.

The reformation function rearranges learned knowledge with respect to each predicate symbol appearing in the head part of Horn clauses, as follows:

- *Step 1:* Delete duplicate knowledge.
- *Step 2:* If there are pieces of background knowledge and hypothesis knowledge with respect to a predicate symbol in the head part, we treat them separately. (While it seems that hypothesis knowledge is not necessary in this case for obtaining a minimal solution hypothesis set, hypothesis knowledge becomes necessary since the generalized version of learned background knowledge does not always lead to a successful proof. We need, therefore, to store hypothesis knowledge even in this case.)
- *Step 3:* Merge the pieces of learned knowledge with the same predicate symbol as their heads into one piece of knowledge, as follows:
    - *Step 3.1:* If there exists Horn clause knowledge with 'fail' in its body part, then, by checking the values of arguments of its head predicate, place the negation of this value combination in the beginning position of the body of reformed Horn-clause knowledge.
    - *Step 3.2:* Extract common fact-type literals in the body parts of different pieces of knowledge, and

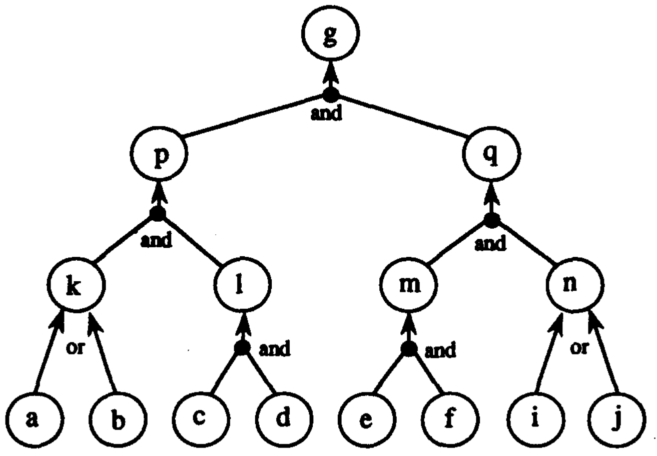

Figure 4 Example of original knowledge

place them in the preceding position of the body of reformed Horn-clause knowledge. Place other uncommon literals appearing in different body parts of originally learned Horn-clause knowledge in latter positions of the body of a reformed Prolog clause with an OR relation.

As an illustration, suppose that the system has the following pieces of learned knowledge:

$g(a, b)$:- fail.
$g(X, Y)$:- $p(X)$, $q(X,Y)$, $r(Y)$.
$g(X, Y)$:- $p(X)$, $s(X,Y)$, $r(Y)$.

Then, these pieces of knowledge are reformed into one Prolog clause in the form of

$$g(X,Y) \text{ :- } \backslash + (X{=}{=}a,\ Y{=}{=}b), p(X), r(Y), (q(X,Y); s(X,Y)).$$

where $\backslash +$ denotes 'not provable', and $(q(X,Y)\,;\,s(X,Y))$ means $q(X,Y)$ or $s(X,Y)$ in Prolog.

To see the effect of the above knowledge reformation mechanism, we consider an example knowledge base with the knowledge structure shown in *Figure 4*. If the experience-based learning takes place several times at the top goal node g of *Figure 4*, the system can acquire the learned knowledge shown in *Figure 5* for g. Suppose, for example, that only the proof of a predicate i fails owing to the unification condition of attached variables, then the system can eventually find a successful proof after checking a twice, b once, c, d, e and f each three times, and j once in the case shown in *Figure 5*. This is very inefficient. Our knowledge reformation mechanism transforms the knowledge structure of *Figure 5* into the structure shown in *Figure 6*. Then, the system can find a successful proof by checking each c, d, e, f, a, i and j only once. Furthermore, when the proof of f fails, the system finds that the set of learned knowledge is useless in this case after checking a and b once, and c, d, e and f each four times, in the case of the unreformed knowledge shown in *Figure 5*. However, using the reformed knowledge in the form shown in *Figure 6*, the system can find the uselessness of the learned knowledge by checking each c, d, e and f only once.

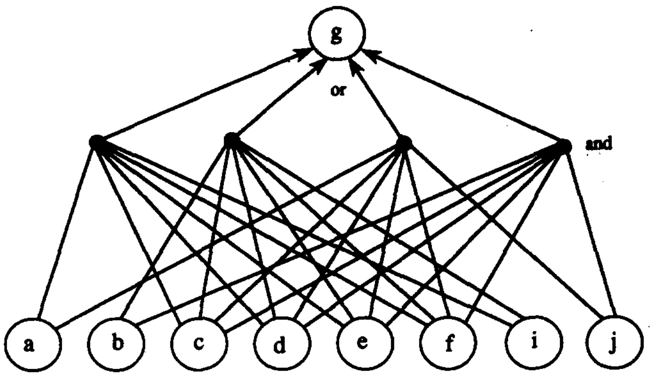

Figure 5 Learned knowledge

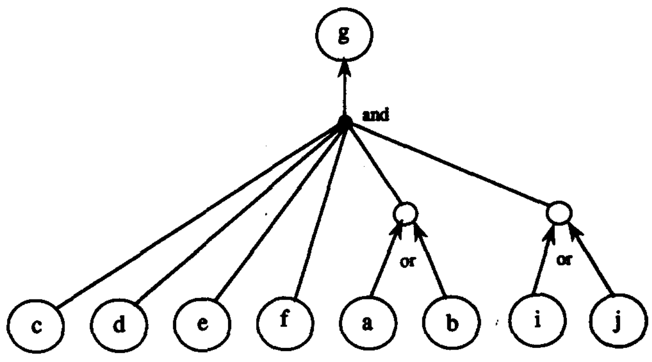

Figure 6 Reformed learned knowledge

The knowledge reformation function allows, as we have seen above, the reduction of inference overheads caused by the addition of learned knowledge, and it contributes to achieving the effectiveness of our experience-based learning.

EXPERIMENTAL PERFORMANCE EVALUATION

We evaluate the performance of our experience-based learning mechanism with the selective learning function at proper subgoals and the reformation function of learned knowledge, using test knowledge sets which are designed artificially to illustrate clearly the learning effect.

Performance evaluation of selective learning at proper subgoals

In this case, the test knowledge sets exemplified in *Figure 7* are used, where left and right subtrees rooted at $p_1(X)$

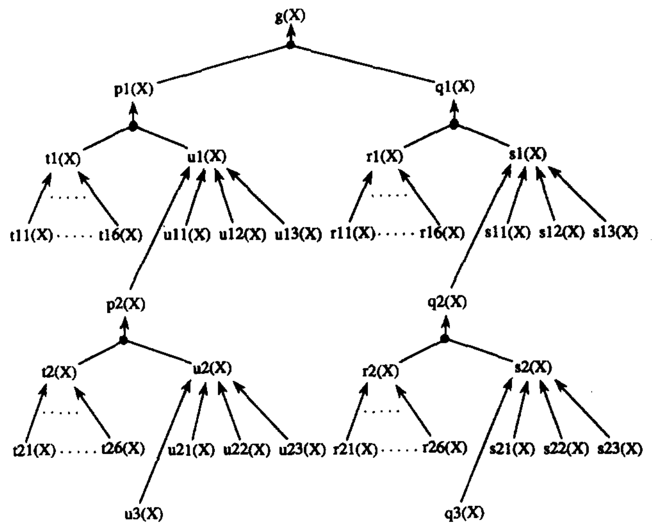

Figure 7 Test knowledge set for evaluating speedup by selective experience-based learning at proper subgoals
[Inconsistency constraints: inc :- *rij*(*X*) *sij*(*x*) {$i = 1, 2, \ldots, j = 1, 2, \ldots$}, inc :- *rij*(*X*), *skl*(*X*) {$i, k = 1, 2, \ldots, (i \neq k); j = 4, 5, 6; l = 1, 2, 3$}. Defined element hypotheses: $ri_1(a)$, $ri_2(b)$, $ri_3(c)$, $ri_4(a)$, $ri_5(b)$, $ri_6(c)$, $si_1(a)$, $si_2(b)$, $si_3(c)$, $ti_1(a)$, $ti_2(b)$, $ti_3(c)$, $ti_4(a)$, $ti_5(b)$, $ti_6(c)$, $ui_1(a)$, $ui_2(b)$, $ui_3(c)$, {$i = 1, 2, \ldots$}.]

and $q_1(X)$, respectively, are the same, except for inconsistency constraints for the left subtree. Also, the subtrees of $q_1(X)$, $q_2(X)$, ... have an identical structure, as do the subtrees of $p_1(X)$, $p_2(X)$, ... We can change the tree depth of the knowledge set to define the scale of knowledge bases or the number of possible element hypotheses. For example, $r_{14}(a)$ and $s_{11}(a)$ in the right subtree become the components of a solution hypothesis set for a given goal $g(a)$ in *Figure 7*. A number of backtrackings take place in the right subtree to find a solution hypothesis first time, because of associated inconsistency constraints.

According to our selective experience-based learning algorithm, $q_1(X)$, $q_2(X)$, ... in the right subtree of *Figure 7* will be selected as proper target subgoals for learning. The experience-based learning for the knowledge set of *Figure 7* is evoked after a goal $g(a)$, $g(b)$ or $g(c)$, for example, is given to the system to acquire effective generalized knowledge. To evaluate the speedup obtained by this experience-based learning at proper subgoals, we also carried out experiments of learning at other subgoals. The learning at $p_1(X)$, $p_2(X)$, ... in the left subtree of *Figure 7* has also been carried out for this purpose.

Figure 8 shows the results of the experiments, which are measured in terms of the required CPU time for finding a solution hypothesis set for a goal, for example $g(a)$, $g(b)$ or $g(c)$. In *Figure 8*, we can clearly see a speedup effect from our selective experience-based learning at proper subgoals. The learning at other than the proper effective subgoals cannot yield speed improvements, because the inferences for these subgoals were initially sufficiently efficient.

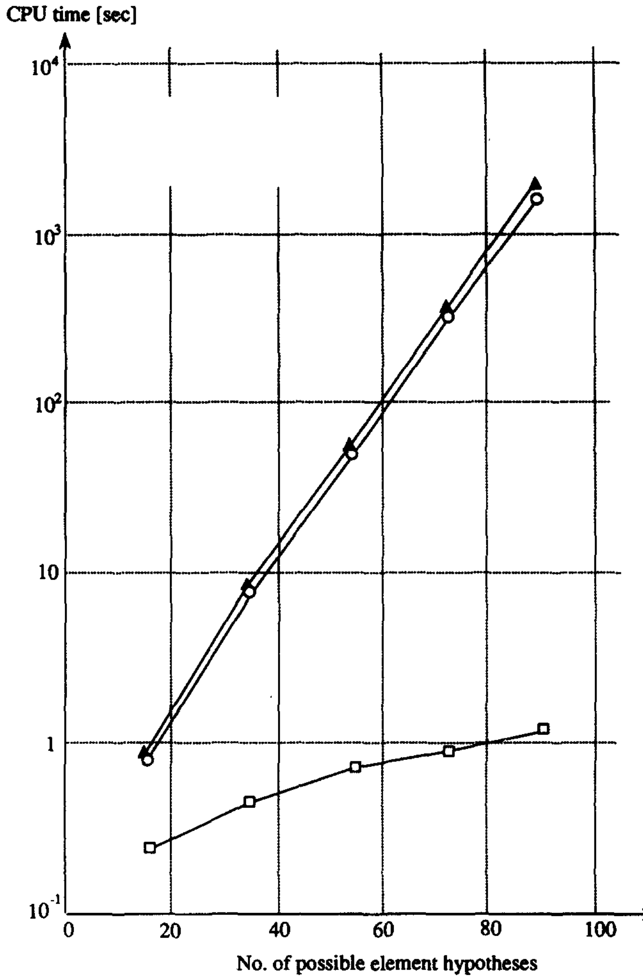

Figure 8 Example of speedup of hypothetical reasoning by selective learning at proper subgoals
[○: without learning, □: with learning at proper subgoals, ▲: with learning at other subgoals.]

Performance evaluation of knowledge reformation

For this evaluation, we used the test knowledge sets exemplified in *Figure 9*, in which the scale of the test knowledge set can be changed by defining the depth of the inference tree. *Figure 9* is a particular case in which the depth is 3. $si_1(V)$, $si_2(V)$, $si_3(V)$, $si_4(V)$ and $si_5(V)$ {$i = 1,2,3$} have the element hypotheses of $si_1(a)$, $si_2(b)$, $si_3(c)$, $si_4(d)$ and $si_5(e)$, respectively, as their child nodes. Inconsistency constraints are defined as

$$\text{inc} :- si_1(V), sj_5(W). \{i,j = 1,2,\ldots,5 \text{ and } i \neq j\}$$

Because of these inconsistency constraints, the subgoals of $q_0(V,W,X,Y,Z)$ and $q_1(W,X,Y,Z)$ in *Figure 9* become the targets of our selective experience-based learning.

We consider the situation in which this learning has taken place thoroughly at these target subgoals; this is the situation after every possible instantiation of $q(V,W,X,Y,Z)$ is given as a goal to the system. In this situation, the number of pieces of learned Horn-clause

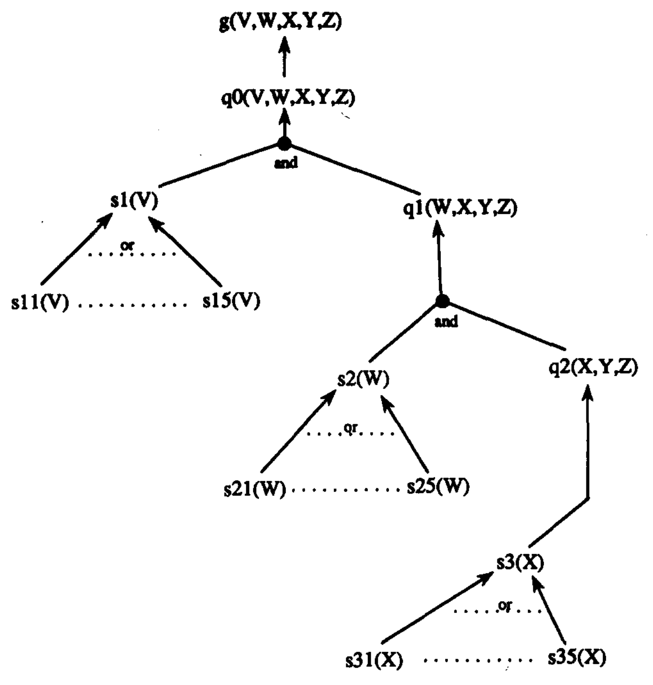
g(V,W,X,Y,Z)
q0(V,W,X,Y,Z)
and
s1(V)
or
s11(V) .... s15(V)
q1(W,X,Y,Z)
and
s2(W)
or
s21(W) .... s25(W)
q2(X,Y,Z)
s3(X)
or
s31(X) .... s35(X)

Figure 9 Test knowledge set for evaluating effect of knowledge reformation
[Inconsistency constraints: inc: $si_1(V)$, $sj_5(W)$ $\{i, j = 1, 2, 3, 4, 5$ and $i \neq j\}$. Defined element hypotheses: $si_1(a)$, $si_2(b)$, $si_3(c)$, $si_4(d)$, $si_5(e)$ $\{i = 1, 2, 3\}$.]

knowledge having $q_0(V,W,X,Y,Z)$ as the head, for example, becomes the third power of 5 minus the number of inconsistent hypothesis combinations, since there are five OR-related child nodes for $s_1(V)$, $s_2(W)$ and $s_3(X)$ in *Figure 9*. In the case of *Figure 9*, the number of these learned pieces of knowledge becomes 99 for $g_0(V,W,X,Y,Z)$. This learned knowledge added to the knowledge base may cause inefficiency unless the knowledge reformation mentioned in the sixth section is performed.

Figure 10 shows the experimental results of the worst-case inference time (CPU time) for finding a solution hypothesis set for a given goal, for example $g(e,e,e,e,e)$. In order to obtain a clear experimental result, the given goals in these experiments were chosen such that they did not evoke backtracking owing to the defined inconsistency constraints in the original knowledge base, and their solutions were found after all the possible trials of unification (the worst case) in the learned knowledge base without knowledge reformation. There were a large number of failures of unification in the inflated knowledge base with the learned knowledge, and the consequent efficiency degradation of the learned knowledge base is shown in *Figure 10*.

However, after the reformation of the learned knowledge base, this inefficiency is removed, as seen in *Figure 10*. If the given goals are such that they give rise to backtrackings because of inconsistency constraints in the original knowledge base, the inference efficiency can be improved to a large extent by experience-based learning

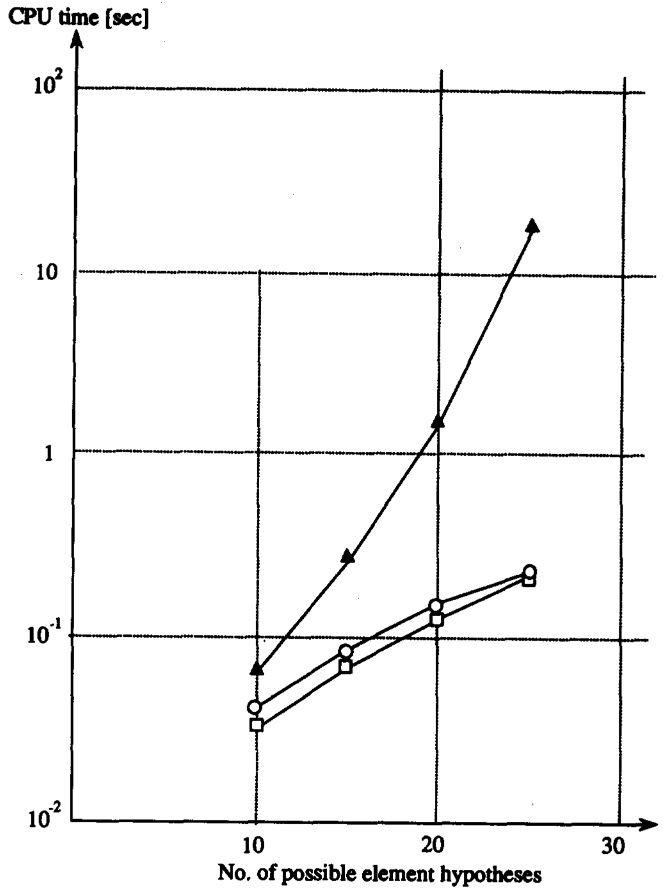

Figure 10 Example of effect of learned-knowledge reformation
[○: original knowledge base, ▲: learned knowledge base without reformation, □: learning knowledge base with reformation.]

and knowledge reformation, which removes the inference overhead caused by the increase in learned knowledge.

CONCLUSIONS

We have presented a hypothetical reasoning system with an experience-based learning mechanism. This learning mechanism acquires generalized knowledge from inference experiences, and enables the subsequent inference processes to be speeded up. In comparison with existing explanation-based learning, the following new functions have been introduced in our experience-based learning mechanism.

- The learning mechanism can work for hypothetical reasoning, where necessary hypotheses are generated and consistency checking among adopted hypotheses is executed in the inference process.
- The learning mechanism can learn effective knowledge even at subgoals that appear in the inference process, and not only at the final goal, so that more widely useful knowledge can be acquired for improving the inference speed in subsequent inference processes.

- In order to avoid similar proof failures occurring more than twice, the learning mechanism learns knowledge from proof failure by deciding whether or not the generalization of acquired knowledge is appropriate, depending on the type of a related inconsistency constraint.
- Since the amount of learned knowledge becomes very large, the system can select effective target subgoals for experience-based learning from the viewpoint of improving the inference speed.
- A reformation function of learned knowledge is provided to reduce the inefficiency caused by the addition of learned knowledge.

The system was applied to a design problem involving digital circuit block synthesis [4], and this showed the speedup effect of inference processes having partial similarity to prior designs. Like explanation-based learning, our experience-based learning is a deductive learning mechanism (for hypothetical reasoning); it thus allows the improvement of inference speed by the reformation of the knowledge base, but it has no power to generate entirely new knowledge. A combination with inductive learning, particularly with inductive logic programming techniques [13], which allow the learning of new generalized knowledge from given positive and negative examples, seems to be one interesting extension of our present system.

REFERENCES

1 Poole, D, Aleliunas, R and Goebel, R 'Theorist: a logical reasoning system for defaults and diagnosis' *in* Cercone, N J and Mccalla, G (Eds.) *The Knowledge Frontier: Essays in the Representation of Knowledge* Springer-Verlag, USA (1987)
2 Poole, D 'A logical framework for default reasoning' *Artif. Intell.* Vol 36 pp 27–47 (1988)
3 Ishizuka, M and Matsuda, T 'Knowledge acquisition mechanisms for a logical knowledge base including hypotheses' *Knowledge-Based Systems* Vol 3 No 2 pp 77–86 (1990)
4 Makino, T and Ishizuka, M 'A hypothetical reasoning system with constraint handling mechanism and its application to circuit-block synthesis' *Proc. PRICAI '90* Nagoya, Japan pp 122–127 (1990)
5 de Kleer, J 'An assumption-based TMS' *Artif. Intell.* Vol 28 pp 127–162 (1986)
6 Ishizuka, M and Ito, F 'Fast hypothetical reasoning system using inference-path network' *Proc. Int. Conf. Tools for AI ICTAI '91* San Jose, CA, USA pp 352–359 (1991)
7 Kondo, A, Makino, T and Ishizuka, M 'Efficient hypothetical reasoning system for predicate-logic knowledge base' *Proc. Int. Conf. Tools for AI ICTAI '91* San Jose, CA, USA pp 360–367 (1991) (and *Knowledge-Based Systems* Vol 6 No 2 pp 87–94 (1993))
8 Kautz, H A and Selman, B 'Hard problems for simple default logics' *Proc. Principles of Knowledge Representation and Reasoning KR '89* Toronto, Canada pp 189–197 (1989)
9 Abe, A and Ishizuka, M 'Fast hypothetical reasoning system using analogy on inference-path network' *J. Japanese Soc. AI* Vol 7 No 1 pp 77–86 (1992) (in Japanese)
10 Mitchell, T M, Keller, R M and Kedar-Carbelli, S T 'Explanation-based generalization: a unifying view' *Machine Learning* Vol 1 pp 47–80 (1986)
11 DeJong, G and Mooney, R 'Explanation-based learning: an alternative view' *Machine Learning* Vol 1 No 2 pp 145–176 (1986)
12 Inoue, K 'Linear resolution for consequent finding' *Artif. Intell.* Vol 56 pp 301–353 (1992)
13 Muggleton S (Ed.) *Inductive Logic Programming* Academic Press (1992)