Proc. of the 1991 IEEE
Int. Conf. on Tools for AI
San Jose, CA-Nov. 1991

Fast Hypothetical Reasoning System using Inference-Path Network

Mitsuru Ishizuka and Fumiaki Ito *
Institute of Industrial Science, University of Tokyo
7-22-1, Roppongi, Minato-ku, Tokyo, 106, Japan

*Presently with Information System Laboratory, Canon Ltd.

Abstract

While it is a very useful knowledge-processing framework applicable to many practical problems, the most crucial problem of logic-based hypothetical reasoning system is its slow inference speed. This paper describes a fast hypothetical reasoning system named KICK-SHOTGAN, which avoids inefficient backtracking by forming a compiled inference-path network followed by the forward synthesis of necessary hypothesis combination along this network. The formation of the inference-path network is based on a linear-time algorithm for the satisfiability testing of propositional Horn clauses. This system differs from ATMS mainly in its total problem solving nature. That is, it works for the logical problem-solving framework which yields a solution for a given goal, whereas the ATMS calculates possible data supported by hypotheses incrementally in response to the input of a justification (rule) from a problem solver existing outside the ATMS. Experimentally, the inference speed of this fast hypothetical reasoning system is thousands of times faster than that of existing systems implemented in Prolog.

1. INTRODUCTION

The handling of incomplete knowledge in the knowledge-base is an important approach for broadening the capability of the knowledge base [1]. Incomplete knowledge means here knowledge which is not always true; more specifically, it means knowledge with exceptions, knowledge with inconsistency, partially missing knowledge, over-generalized knowledge, etc. In general, a non-monotonic reasoning system is required to handle incomplete knowledge in the knowledge-base.

A logic-based hypothetical reasoning system [2,3], which can deal with incomplete knowledge as hypothesis, is a useful framework because of its theoretical basis and its applicability to practical problems including diagnosis [2,4] and design [5]. Its formalism has a close connection with constraint satisfaction problem (CSP). The most crucial problem of this reasoning system, a form of non-monotonic reasoning system, is its slow inference speed. An immediate remedy for this problem is to incorporate heuristic knowledge which plays the role of guiding the inference. However, it is difficult to cover the whole problem domain by heuristic knowledge, which causes the well-known knowledge acquisition problem. Therefore, we have to develop a fast inference mechanism not relying on heuristic knowledge from the viewpoint of finding a solution under the logically described constraints.

By analyzing the behavior of the hypothetical reasoning system implemented using the mechanism of Prolog, we consider that the backtracking caused by the inconsistency among selected hypotheses is the major factor of deteriorating the inference speed. Then, we present a two-phase hypothetical reasoning system [6], where a goal-directed inference-path network is formed using the complete knowledge set but excluding hypotheses in first phase. Hypothesis sets necessary for proving a given goal are synthesized in the second phase along this inference-path network, in a forward inference fashion with no backtracking. The inference-path network also allows to reduce the number of computationally expensive hypothesis combination to a minimum. The formation of the inference-path network is based on a linear-time algorithm for the satisfiability testing of propositional Horn formulae [7]. Experiments show that this fast hypothetical reasoning system can achieve an inference speed more than 1,000 times faster than that of Prolog-based implementation.

2. LOGIC-BASED HYPOTHETICAL REASONING SYSTEM

The hypothetical reasoning in this paper is a logic-based one [2,3], where knowledge is divided into two categories, i.e., complete knowledge (or fact in [2,3]) and hypothesis. Complete knowledge denoted as F is knowledge which is always true and has no possibility of

0-8186-2300-4/91 $01.00 © 1991 IEEE

inconsistency. On the other hand, the hypothesis denoted as H is incomplete or defeasible knowledge for which consistency checking is required in the inference process.

The basic behavior of this hypothetical reasoning is as follows. When a goal (or an observation) G is given, the system first tries to prove the goal from complete knowledge. If it failes, then the system selects a subset of the hypotheses so that the given goal is proved from the union of complete knowledge and this hypothesis subset. The selected subset of the hypotheses should be consistent with complete knowledge, while inconsistency is allowed in the whole set of the hypotheses. In ordinary logic-based problem solving, the success or failure of deductive proof becomes the answer. When the goal includes variables, the binding (unification) to the variables becomes an answer in the success case. On the other hand, a selected subset of hypotheses becomes an answer in the logic-based hypothetical reasoning system, in which the deductive inference mechanism is utilized in reverse direction to generate a solution hypothesis subset. This generative nature allows model-based problem solving in the areas of diagnosis, design, etc. When compared with a production system, this system inherits the good properties of logial precise semantics.

The structure of the above hypothetical reasoning system can be summarized to find a solution h of

- $h \subseteq H$ (h is a subset of H),
- $F \cup h \vdash G$ (G can be derived from $F \cup h$), and
- $F \cup h \nvdash \square$ ($F \cup h$ is consistent, $\square$: empty clause),

where F, H and G are the complete knowledge, possible hypotheses and a given goal, respectively. In addition, it is often required for the solution hypothesis subset h to be a minimal subset; that is, no subset h' of h satisfies the above conditions.

This hypothetical reasoning system can be constructed on first-order predicate logic. We restrict, however, the knowledge representation to Horn clauses so that fast inference can be achieved. Furthermore, in this paper, we only deal with the hypothetical reasoning system represented in propositional Horn clauses with no variable, since our main concern here is the fast inference mechanism. (The representation of first-order predicate logic with no function can be expressed in propositional logic in the Herbrand domain.) Since the logical negation of an atom cannot be expressed in Horn clauses, we introduce an atom called "inconsistent" to denote inconsistency among hypotheses, such as,

inconsistent : -*h1*, *h2*.

which says that *h1* and *h2* cannot be coexist in an environment. This expression is also useful for users to denote inconsistent relations.

3. Problems with the Implementation using the Inference Mechanism of Prolog

The logic-based hypothetical reasoning system can be implemented easily by using the inference mechanism embedded in Prolog. In this case, the inference proceeds in a backward fashion starting from a given goal as in Prolog. This is a goal reduction process in which the system tries to reduce the goal to empty by adopting available knowledge. In the first step, a hypothesis-box (h-box) which stores adopted hypotheses is set to be empty. As the inference proceeds, the system adopts a new hypothesis from the hypothesis base when the reduction of the goal or a derived subgoal can not be extended with using only complete knowledge and hypotheses existing in the h-box. This new hypothesis is put into the h-box only when it is consistent with the hypotheses already existing in the h-box and complete knowledge. This consistency is checked by the failure of proving the 'inconsistent' atom using the adopted hypotheses and complete knowledge. If no

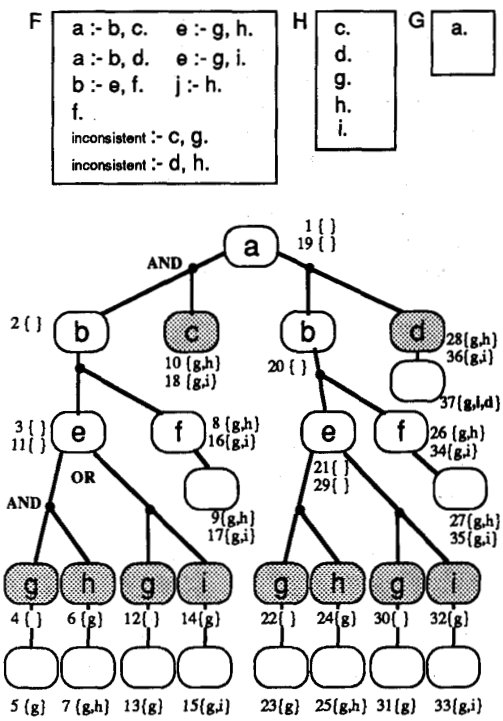

Fig. 1 An example of hypothetical reasoning on Prolog. (The number attached to the node indicates the search order, and { } shows adopted hypotheses. In this case, the goal 'a' can be proved by adopting the hypothesis {g,i,d}.)

appropriate hypothesis exists, then the backtracking is invoked to search other remaining reduction branches. In this case, the adopted hypotheses between the backtracked tip node and the returned node are discarded from the *h*-box. This inference process is repeated until the goal reduction succeeds or no available hypothesis exists. If it is successful, the adopted hypotheses existing in the *h*-box become a solution for proving the given goal. (All the solution hypotheses can be found by a forced backtracking of all the possible branches.) Figure 1 illustrates this inference process, where $\{g, i, d\}$ can be obtained after backtracking as a solution hypothesis for proving a given goal '*a*'.

This type of simple implementation has a severe problem of slow inference speed. The inefficiency of the inference comes from the following causes.

1) Hypothesis adoption and associated consistency checking is executed even for a branch which eventually fails due to the lack of appropriate complete knowledge and possible hypothesis.
2) Upon the change of adopted hypothesis due to backtracking, the search of subtrees irrelevant to this hypothesis change are also executed. (For example, in Fig.1 the subtree below the subgoal '*f* ' is searched every time in spite of the fact that the inference in this subtree is irrelevant to the hypothesis change in other nodes. Although this is a general problem in backward inference with backtracking, the problem is severe in hypothetical reasoning since the backtracking is invoked by the inconsistency among the adopted hypotheses as well.)
3) The same search branch (subtree) may appear more than once. (This is also a general problem with Prolog. In Fig.1, the subtree below the subgoal '*b*' is searched twice.)
4) Supersets of the hypothesis set found already as inconsistent may be searched again. (Since the adoption of hypothesis depends on the derivation tree structure, it is not possible to avoid the adoption of these supersets. In Fig.1, an inconsistent combination of hypotheses $\{c, g\}$ is generated twice.)
5) The consistency checking is expensive since it is executed by the resolution-type proof procedure.

Although a part of the above-described problem is due to the inefficiency of Prolog itself, the problem becomes severe in hypothetical reasoning since backtracking occures frequently due to inconsistency among adopted hypotheses. Therefore, a first key point of improving the efficiency of the hypothetical reasoning is to avoid inefficient backtracking due to inconsistency among hypotheses. This can be realized in general by using parallel forward reasoning. Simple forward reasoning, however, generates a large number of inference branches or intermediate nodes, most of which are irrelevant to the proof of a given goal. Thus it is necessary to identify hypotheses relevant to the proof of the goal, and then execute inference concerning only these hypotheses.

The adopted hypotheses are combined at intermediate nodes, where the inconsistency check and the deletion of hypothesis sets subsumed by another set are also executed. This processing is expensive. Thus, a second key point of improving the efficiency is to reduce expensive hypothesis combination to a minimal number.

Based on these considerations, we have developed a fast hypothetical reasoning system named KICK-SHOTGAN (*Knowledge-base handling Incomplete Knowledge - by Synthesizing Hypotheses through Generated Path on Network*), in which an inference-path network plays an important role in achieving fast inference. All the above-mentioned problems have been solved in this system.

4. Fast Hypothetical Reasoning using Inference-Path Network

The purposes of forming a inference-path network are 1) to restrict the forward inference branches only to goal-directed ones, and 2) to reduce hypothesis combination processing to a minimum. The formation of this network is based on a linear-time algorithm for testing the satisfiability of propositional Horn clauses [7]. In other words, this linear-time algorithm is employed as a pre-processing step for reducing the computational cost of expensive hypothesis combination processing.

Rule-type fypotheses are allowed in the KICK-SHOTGAN. They are, however, transformed as a pre-processing into newly introduced single-atom hypotheses and modified complete knowledge. For example, a rule-type hypothesis "*a:-b.*" is transformed by introducing a new atom "*c*" into,

complete knowlegde *a:-b,c.*, and
hypothesis *c.*

According to this pre-processing, all the hypotheses become unit clauses (single atoms), which positions in the inference network become leaf nodes.

The KICK-SHOTGAN consists of the following two phases.

4.1 Inference-Path Network Formation Phase (Phase-1)

In this phase-1, the system first constracts a goal-directed inference-path network using only complete knowledge excluding hypotheses, which are afterward associated with the leaf nodes of the network. Since the process of this phase-1 is based on the application of the linear-time algorithm for testing the satisfiability of propositional Horn clauses [7], we describe the process

according to the method of [7].

The nodes of the network correspond to the atoms appearing in complete knowledge. The same atom appearing in different clauses is treated as one node; therefore, the inefficiency of recalculating the same subtree in the Prolog-based implementation can be avoided. The merging of the same atom into one node in the inference network is possible only in the propositional-logic case with no variable.

We assume that the goal is represented, for example, as,

goal : - *a*, *b*, *c*.

where the goal is satisfied if *a*, *b* and *c* are true. In addition to this 'goal' node and the 'inconsistent' node, we introduce an additional 'true' node so that a uniform treatment can be possible for all knowledge sentences. The directed links between the nodes are set up depending on the types of clauses as follows.

- Rule-type clause

  A directed link is established from each atom in the body part of the clause to the atom in its head part. A unique number is assigned to the links corresponding to the same clause. The links with the same number represent an AND relation. As the goal clause can be regarded as a rule-type clause, this type of directed link is established for the goal clause as well.

- Fact clause

  The fact clause of complete knowledge is a unit clause declaring a fact which is true all the time. A directed link is established from the 'true' node to the node corresponding to the atom of the fact clause.

- Clause with 'inconsistent' head.

  A directed link is established from each atom in the body part of the clause with the head of 'inconsistent' to 'inconsistent' node. A unique number is assigned to the links corresponding to the same clause.

The following initial states are assigned to the nodes;

"true" state to 'true' node,

"true-by-hypothesis (true-by-*h*, in short)" state to hypothesis node,

"false" state to other nodes.

As described later, the node with the "true-by-*h*" state holds a hypothesis-box (*h*-box) to store the sets of hypotheses (called environments in ATMS [8]) necessary for supporting its "true-by-*h*" state. (The *h*-box in our system corresponds to label in ATMS.) The hypothesis set is represented internally in a computer as a bit-vector as in the ATMS [8] for efficient handling of hypothesis combination and consistency checking.

If all the tail nodes of the same numbered directed

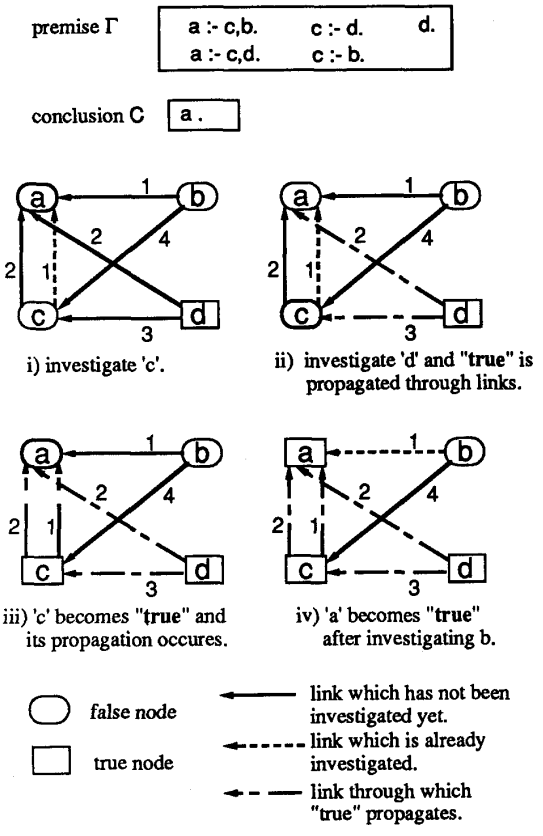

Fig. 2 An example of goal-directed inference by "true" state propagation on network.

link are in the "true" state, then the "true" state can be propagated to the tip node by changing its state to "true". This propagation is repeated until no further "true" state can be generated. If the "true" state is propagated to the 'inconsistent' node, then it turns out that the inconsistency exists in the complete knowledge *F*; this situation should be avoided by deleting inappropriate knowledge. If the 'goal' node does not have "true" state at this moment, it can not be satisfied by only using complete knowledge; and consequently the adoption of consistent hypotheses is required.

The propagation of the "true" state to a certain node can be executed by a goal-directed search as illustrated in Fig. 2, where the 'true' node is omitted by setting the initial state of the fact clause node (*d* in this case) to "true". A compiled inference-path network for a given goal can be formed by using this type of goal-directed propagation of the node state. In addition to the "true" state propagation, the "true-by-*h*" state is also propagated if all the tail nodes

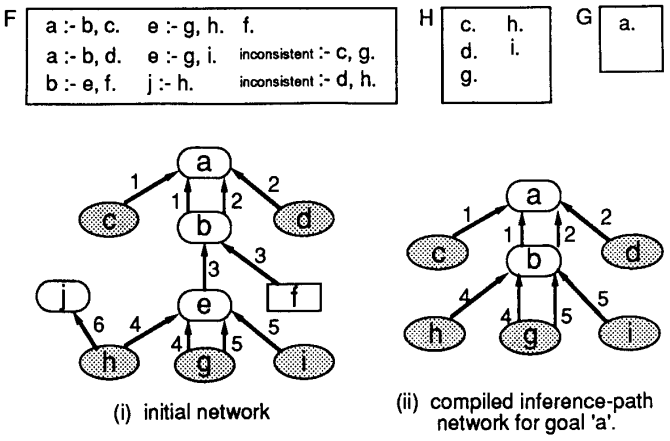

Fig.3 An example of inference-path network formation.

of the same numbered directed links are in the 'true-by-*h*' or 'true' state. At this moment, computationaly expensive hypothesis propagation and combination is not executed and postponed to phase-2. The "true" state assignment can overwrite the "true-by-*h*" state. Starting from the goal node, the system can determine the state of the connected nodes in a backward inference manner.

The hypothesis which contributes to prove the goal is only synthesized from the nodes with "true-by-*h*" state. Only these nodes should be considered in synthesizing the necessary hypotheses in the phase-2; thus, the system extracts this portion from the network. A pair of two tip and tail nodes is merged into one node if the tip node has only one incoming link and the tail node has only one outgoing link. The inference-path network for the goal is thus formed. Figure 3 shows the inference-path network formation for the same example as in Fig. 1. In Fig. 3, '*j*' node is removed since it is irrelevant to the goal, '*f*' node is also removed since it has 'true' state which does not have any influence on the propagation of the hypothesis. Also, '*e*' and '*b*' nodes are merged into one node. The inference-path network allows the minimization of computationally expensive hypothesis combination through the network.

4.2 Hypothesis Synthesis Phase (Phase-2)

The node with the "true-by-*h*" state becomes true supported by hypothesis when there exists more than one hypothesis set in its *h*-box. In the beginning, all the *h*-boxes are set to empty. A simple parallel forward inference without backtracking can be executed by placing hypotheses into their corresponding leaf nodes in the inference-path network and by propagating them along the directed links. The union or product of the hypothesis sets of lower nodes is synthesized depending on OR or AND relation at upper nodes.

The synthesized hypothesis set is subjected to a consistency check. As in ATMS [8], the system checks whether or not the synthesized hypothesis set is a superset of the inconsistent hypothesis sets (called nogood sets in ATMS) by comparing their bit-vectors. A minimality check is also conducted to remove redundant hypothesis sets. By comparing the bit-vectors of the hypothesis sets existing in the same node, the system removes the redundant hypothesis sets subsumed by another hypothesis set. After theses checks, the consistent and non-redundant synthesized set is placed into the *h*-box.

The simple execution of consistency and minimality checks described above is not necessarily efficient, since these checks are executed every time against the inconsistent hypothesis sets and already synthesized hypothesis sets. Thus, the notion of stage and redundancy-marker has been introduced in KICK-SHOTGAN to improve the efficiency of these checks.

Each stage *S* is characterized by a corresponding element hypothesis *hs*. The stage starts from putting one hypothesis into the *h*-box of the corresponding leaf node. This hypothesis is propagated through the directed links as long as possible. If further propagation becomes impossible, then one stage ends. Only one hypothesis is inserted into the network's leaf node in one stage. Accordingly, hypothesis sets synthesized in this stage *S* always include the element hypothesis *hs*.

The hypothesis sets synthesized before this stage *S* never include this element hypothesis *hs*. Therefore, these hypothesis sets can never be redundant to the new

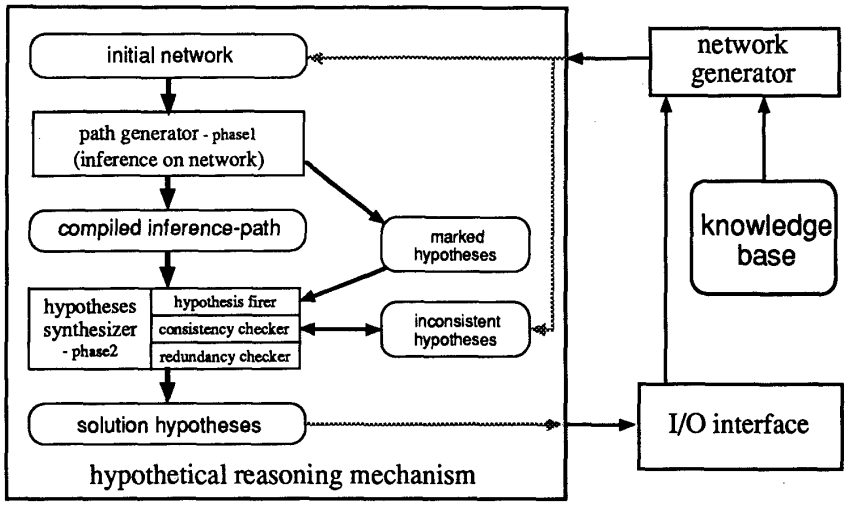

Fig.4 System configuration of KICK-SHOTGAN.

hypothesis set synthesized in the stage *S*. Accordingly, the minimality check can be omitted for the case of the above-mentioned relation.

The inconsistent hypothesis sets are found when the hypothesis sets put in the *h*-box of the 'inconsistent' node. The system removes these sets and these supersets from all the *h*-boxes except one of the 'inconsistent' node. An inconsistent hypothesis set found at a stage *S* always contains an element hypothesis *hs* characterizing the stage *S*. The hypothesis sets which should be removed from the *h*-boxes also contain this element hypothesis *hs*, since they are the supersets of the inconsistent hypothesis set found at the stage *S*. Therefore, the inconsistency check with respect to the inconsistent hypothesis set found at a stage *S* can be restricted only to the hypothesis sets synthesized at the same stage *S*; no other checks are necessary.

In the present system, the sequence of hypothesis placement into the network is determined according to the order of hypothesis description in the hypothesis base. There may be a more efficient method for this sequence determination.

A redundancy marker has been introduced after a study about the case that redundant hypothesis sets are generated. The redundant hypothesis set is generated only when a component of this set is used at a lower node to synthesize more than two hypothesis sets. This case occurs only at a node having more than two outgoing links. The hypothesis set which is synthesized via nodes having only one outgoing link never make other hypothesis sets redundant, nor can it be made redundant by other sets. A redundancy marker IND (abbreviation of independent) is assigned to this type of hypothesis set. The hypothesis set which is synthesized via nodes having more than two outgoing links has the possibility of redundant relation with another set. A redundancy marker DEP (abbreviation of dependent) is set to this type of hypothesis sets. The hypothesis set synthesized from the two hypothesis set having DEP and IND markers may become redundant with respect to another set, but never make other hypothesis sets redundant. A redundancy marker SUP (abbreviation of superset) is set to this type of hypothesis set. These redundancy markers IND, DEP and SUP are assigned to the hypothesis sets when they are synthesized. The system can reduce the cost of the minimality check by using the information of these redundancy markers attached to the hypothesis sets.

Although a part of the behavior in phase-2 is similar to the label updating in the ATMS [8], the system of this paper differs from the ATMS in that 1) the synthesis and propagation of the hypothesis sets are guided by the compiled inference-path network, and 2) the efficient checking of the consistency and minimality of the hypothesis set is achieved by introducing the stage and three redundancy markers.

Figure 4 shows the system structure of KICK-SHOTGAN incorporating these inference mechanisms.

5. EVALUATION OF INFERENCE SPEED

The speed improvement of KICK-SHOTGAN has been evaluated by using test examples. The examples here are taken from the fault diagnosis problems of logic

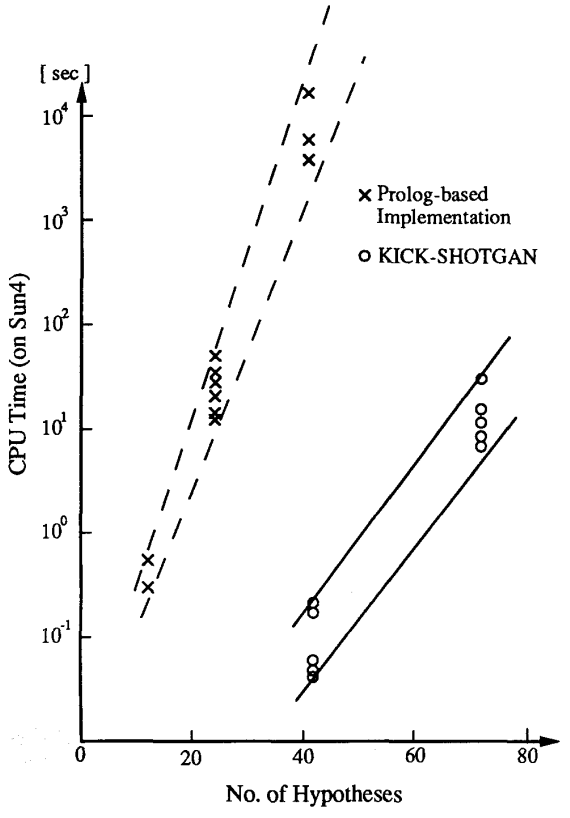

Fig.5 Inference speed (CPU time on Sun4) v.s. number of possible hypotheses in fault diagnosis problems.

circuits, where possible faults of each gate are described as hypotheses. By changing the scale of logic circuits, the different sizes of hypothetical reasoning problems are generated. The inference speeds have been measured in CPU time running on SUN4/260. The Prolog-based version of the hypothetical reasoning has been implemented in Sicstus Prolog. After the compilation of this Prolog program, its execution program is generated. The KICK-SHOTGAN has been implemented in *C*. After compilation by *gcc* compiler on SUN4, the execution program was obtained.

Figure 5 illustrates the measured inference speeds for finding all the possible solutions. In an inference mode of finding a single solution, a faster inference speed can be obtained, depending on the structure of search tree and the location of a solution in the tree. The measurements for evaluation have been made for the cases of finding all the solutions, since they reflect the effect of search space pruning well. The KICK-SHOTGAN can achieve the speed more than 1,000 times faster than that of the Prolog-base implementation in several evaluation tests including the examples shown in Fig. 5.

6. CONCLUSION

A fast logic-based hypothetical reasoning system has been presented. A large improvement (more than 1,000 times faster) of inference speed has been achieved by employing the inference-path network. Key points for this achievement are the avoidance of backtracking due to the inconsistency among adopted hypotheses and the reduction of the number of computationally expensive hypothesis syntheses. An additional improvement has been achieved by introducing the stage concept in the hypothesis adoption and the redundancy marker.

This system differs from the ATMS [8] mainly in its total problem solving framework. That is, this system works to yield a solution for a given goal based on a logical problem solving framework. In other words, this systems searches a solution by satisfying the logically described constraints. On the other hand, the ATMS calculates possible data supported by hypotheses in response to the input of a justification (rule) from a problem solver (production system in most cases) existing outside the ATMS. In the ATMS, it is left to users to guide the inference direction to reach a goal by writing appropriate rules.

Although a large improvement of the inference speed has been achieved in this system, the computational cost still remains exponential order with respect to the number of possible hypotheses as seen in the logarithmic scale of vertical axis in Fig. 5. The computational complexity of non-monotonic reasoning including hypothetical reasoning has been proved as NP-complete or NP-hard [9,10]. Thus, the wall of exponential computational cost cannot be overcome as long as we stay in search methods in ordinary sense. We have found an efficient mechanism of using analogy to overcome this speed limit in the sense of average inference time (not worst-case time) [11]. The inference-path network introduced in this paper plays an important role also in realizing an efficient analogical inference mechanism.

Although only propositional Horn clauses are permitted as knowledge representation in this paper, a similar concept for fast inference can be applicable to the knowledge represented in predicate Horn clauses with variables [12]. However, in the case of predicate clauses, literals with the same predicate and the same constant arguments cannot be treated as one node in the inference-path network if they have variables as arguments, while the same atom appearing in different clauses are merged into one node in the case of propositional clauses.

Acknowledgments

The authors are grateful to Prof. Randy Goebel (Univ. of Alberta) for his disscussion and coments on this paper. This work was supported by the Grant-in-Aids of Ministry of Education, No.0245152(B) and No.02215105 (Special Area on Intelligent Info. and Communications).

References

[1] M. Ishizuka, "An Approach to Next-generation Knowledge-base Systems by Handling Incomplete Knowledge (in Japanese)", Jour. JSAI, Vol. 3, No. 5, pp. 552-562 (1988).

[2] D. Poole, R. Aleliunas and R. Goebel, "Theorist : A Logical Reasoning System for Defaults and Diagnosis", in The Knowledge Frontier : Essays in the Knowledge Representation, (N.J. Cercone and G. McCalla (Eds.)), Springer-Verlag, N.Y. (1987).

[3] D. Poole, "A Logical Framwork for Default Reasoning', Artif. Intelli., Vol.36, pp.27-47 (1988).

[4] M. Ishizuka and T. Matsuda, "Knowledge Acquisition Mechanisms for a Logical Knowledge Base including Hypotheses", Knowledge-Based Systems, Vol. 3, No. 2, pp. 77-86 (1990).

[5] T. Makino and M. Ishizuka, "A Hypothetical Reasoning System with Constraint Handling Mechanism and its Application to Circuit-Block Synthesis", Proc. PRICAI'90, pp. 122-127, Nagoya (1990).

[6] F. Ito and M. Ishizuka, "A Fast Hypothetical Reasoning System utilizing Logical Constraints (in Japanese)", Tech Report of AI Research Group, IPS of Japan, AI70-5 (1990).

[7] W.F. Dowling and J.H. Gallier, "Linear-time Algorithm for Testing the Satisfiability of Propositional Horn Formulae", Jour. of Logic Program., Vol. 3, pp. 267-284 (1984).

[8] J. deKleer, "An Assumption-based TMS", Artif. Intelli., Vol. 28, pp. 127-162 (1986).

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

[10] T. Bylander, D. Allemang, et. al., "Some Results Concerning the Complexity of Abduction", ibid.

[11] A. Abe and M. Ishizuka, "Fast Hypothetical Reasoning System using Analogy on Inference-path Network (in Japanese)", Tech Report of AI Research Group, IPS of Japan, AI72-2 (1990).

[12] A. Kondo and M. Ishizuka, "An Efficient Hypothetical Reasoning System for Knowledge-base represented in Predicate Logic", ibid, AI73-3 (1990), Also Proc. 3rd TAI, San Jose (1991).