TR-379

Problem Solving with Hypothetical Reasoning

by

K. Inoue

May, 1988

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191 ~ 5
Telex ICOT J32964

Institute for New Generation Computer Technology

Problem Solving with Hypothetical Reasoning

Katsumi Inoue

ICOT Research Center,
1-4-28, Mita, Minato-ku, Tokyo 108, Japan

csnet: inoue%icot.jp@relay.cs.net
uucp: {kddlab,mit-eddie,ukc,enea,inria}!icot!inoue

Length of paper: approximately 4600 words

Topic: Applications

Subtopic: Knowledge-based systems

Keywords: truth maintenance, abductive reasoning, hypothetical reasoning, problem solving, search, constraint satisfaction, design

Abstract

Reasoning with incomplete knowledge is a powerful technique for advanced reasoning mechanisms which are much more intelligent than present knowledge-based systems. We treat reasoning with incomplete knowledge as a problem solving system with hypothetical reasoning. For existing methods to handle hypotheses, there has not been the consideration from the standpoint of practical use in problem solving. As a whole problem solving system, hypothetical reasoning should be incorporated in the form of processes for the generation, selection and verification of hypotheses; however, each role has not been clarified so far. A hypothetical reasoning system named APRICOT, which is currently being developed, takes such problems into account. The basic concepts of APRICOT are: (1) with domain-dependent knowledge, especially deep knowledge and commonsense knowledge, and with constraints, hypotheses are automatically generated or enumerated, and (2) the maintenance of multiple contexts is regarded as a mechanism of inference control and are considered as a general procedure providing an interface between a domain-independent truth maintenance system and a domain-dependent problem solver. An application of APRICOT to design is also shown to solve constraint satisfaction problems in design.

1. Introduction

Search is an integral element of all AI systems. For instance, synthetic problem solving (such as design and planning) as well as advanced inference mechanisms (such as non-monotonic reasoning, inductive reasoning and analogies), are in essence combinatorial problems; hence, the avoidance of combinatorial explosion is a major problem of problem solving. In past AI systems, problem solving frameworks were considered from the viewpoint of efficiency in search processes. In general, the problem solving process may be represented by a model in which an AND/OR graph is constructed incrementally at the same time that the search is conducted. In executing searches using such a graph, if the production rules are given as powerful heuristics in conventional rule-based systems, then deterministic choices are made for all the OR parts in the graph. In other words, this determinicity in knowledge-based systems has come from *complete knowledge*, which ascertains their truth values based on traditional logic. However, when there are alternatives among items of knowledge in problem solving, so that deterministic choices on OR branches are not possible, or when we want to allow for *incomplete knowledge* such as knowledge with exceptions or tentative knowledge, all possible solutions must be searched, otherwise a single branch must be non-deterministically chosen and reasoning processed further from there. When such an inference does not result in a satisfactory solution, it may be necessary to backtrack and try a different choice or choices. However, present knowledge-based systems are not yet sufficiently capable of such OR part processing. Hence, this problem should be considered in terms of *hypothetical reasoning*, which has recently been the focus of considerable attention.

Hypothetical reasoning is desirable when, in problem solving, there are alternatives between knowledge, or when incomplete knowledge must be utilized. The term refers to inference which proceeds using incomplete or contradictory knowledge, taken as hypotheses. In hypothetical reasoning, since there may be cases in which contradictions occur because of indefinite or unsound knowledge, it is necessary to check *consistency* with constraints in knowledge bases or to modify hypotheses, and some techniques for *non-monotonic reasoning* or *belief revision* must be considered. Hypothetical reasoning is in fact a form of reasoning used by humans, and is one key to realizing advanced inference mechanisms. Recently, the importance of hypothetical reasoning has come to be recognized, and functions for realizing hypothetical reasoning as naturally as possible are awaited. With regard to this hypothetical reasoning, section 2 below discusses research on architecture, and section 3 addresses problems to be considered relating to practical use. Configurations of the system called *APRICOT* (Assumption-based PRoblem solving Interface for COnsistent Theories; or Assumption-based PRoblem solver in ICOT), which takes these matters into account are discussed in section 4, and section 5 looks at applications to design.

2. Frameworks for Hypothetical Reasoning

Past studies on treatments of hypotheses have been conducted from various angles, as indicated below, and numerous methods and technologies have been proposed.

1. **Uncertainty treatment in expert systems:** Expert systems such as MYCIN and EXPERT derive intermediate hypotheses and final conclusions on the basis of heuristic rules. Different hypotheses are weighted by values which represent credibility or preference. Theses systems are plagued by difficulties in establishing the *certainty factors*; also, problems arise in trying to combine multiple certainty factors.

2. **Truth maintenance systems (TMSs):** These are methods of hypothetical reasoning, which is currently the focus of much attention. In contrast with 1 above, many TMSs do not use values to weight hypotheses. Systems such as [Stallman & Sussman 77], [Doyle 79] and [McDermott 83] are classified as *justification-based TMSs* (JTMSs). Such systems preserve the consistency of databases during reasoning processes based on the supporting justifications; however, the form in which hypotheses are given has not been discussed in great detail, and there are also many outstanding problems concerning efficiency. In [de Kleer 86a], a global, concurrent representation of all contexts by labeling each item of data with dependent hypotheses is kept, but is classified as an *assumption-based TMS* (ATMS).

3. **Hypothesis selection by logical inference:** In systems such as [Finger & Genesereth 85], [Poole 86] and [Reiter & de Kleer 87], hypotheses are chosen logically to explain some observations on the basis of *abductive reasoning*, keeping consistency with given knowledge bases. While they appear to be more inefficient than 2 above, it is possible to check consistency in the logical framework, so that further developments in this area are anticipated.

4. **Non-monotonic logics:** Research is underway on *non-monotonic logics* [Bobrow 80], including *Non-monotonic Logic I* which came out through formulation of JTMS, *default logic*, and *circumscription*. In hypothetical reasoning, the addition or deletion of justifications has a non-monotonic effect on the databases; moreover, the role of TMSs as a mechanism for *belief revision* in such non-monotonic reasoning processes has recently come to be emphasized.

All of these areas of research, especially 2 to 4 above, are interrelated, [1] and within a system as a whole should be incorporated in the form of processes for the *generation*,

[1] The relationships between abductive reasoning, default logic, the closed world assumption and TMSs are discussed in [Inoue 88b], where the difference between the ATMS and the JTMS can be

selection and *verification* of hypotheses; however, each area has tended to focus on only part of these three processes. To achieve a comprehensive explanation of problem solving and inference in numerous fields, a single unified framework for hypothetical reasoning is necessary. In APRICOT, which is currently being developed in ICOT, its architecture, described in Section 4, is considered with attention to practical use of hypothetical reasoning (in Section 3) in problem solving.

3. Problems from the Standpoint of Practical Use

There is no definitive system for the practical application of hypothetical reasoning, and many problems must still be addressed. Hence, before discussing the construction of practical systems, it is essential first of all to research theoretical foundations such as [Reiter& de Kleer 87] and [Inoue 88b] and to define well or understand hypothetical reasoning. However, it is also important to seek answers to such questions listed below, as to what knowledge may be hypotheses and what methods or technologies are needed in solving actual problems.

1. *Domain-dependent problem solvers are needed:*

Few current expert systems and tools are provided with functions for hypothetical reasoning; and the simple inclusion of the JTMS and the ATMS leads to the problem of combinatorial explosion. As the basic method, it would be good to have a formulation of hypothetical reasoning in a unified framework, but immense efforts are necessary to make such a scheme capable of handling all the many unsolved problems of interest. Further, it is difficult to apply hypothetical reasoning practically based solely on a general-purpose architecture.

2. *Multiple extension problems are unsolved:*

---

analyzed as the difference between keeping all models and keeping the one best model reflecting the some intended meaning.

When faced with a choice between models in a design process, for instance, several competing possible worlds may exist corresponding to various items of alternative knowledge; hence, the management of multiple contexts is necessary. When there are multiple contexts, which context should be selected is a problem. We believe that this problem should be solved dependently on problem domains, because at present it seems to be too weak for a general framework of commonsense reasoning like non-monotonic logics to select one context [Hanks & McDermott 87].

3. *Other problems:*

In addition to the above observations, it is important to address the following problems and questions concerning the application of hypothetical reasoning.

(1) What kinds of knowledge are possible hypotheses?

(2) How should the order of priority among hypotheses be expressed?

(3) On what basis will hypotheses be judged true or false?

(4) What kinds of knowledge are contradictions or how are contradictions represented? When contradictions occur, in what form will they be stored in the database? How are records of contradictions utilized in subsequent reasoning?

(5) On what level should hypothetical reasoning be supported? Two types of hypotheses are conceivable — hypotheses provided by the user, and hypotheses which are not recognized as such by the user, but which the problem solver regards as hypotheses with respect to functions and efficiency. In the latter case, disjunctive knowledge may all be treated as hypotheses.

(6) When large scale and/or complex problems are handled, hypotheses must be considered at various levels of problem solving. In such cases, it is not natural to treat hypotheses on such different levels as a single combination of hypotheses, as in the ATMS. Moreover, one may want to introduce hypothetical reasoning on some levels, but not on other levels. It is very important for practical applications that

a hierarchical architecture for hypothetical reasoning be considered.

(7) Many TMSs record all dependencies between data, requiring extremely large amounts of memory. Are there methods which would enable the recording only of those dependencies which are necessary?

Section 4.2 shows that APRICOT solves parts of the above problems.

4. Configuration of a Hypothetical Reasoning System

This section describes the APRICOT system for hypothetical reasoning. APRICOT takes problems described in Section 3 into account. The basic concepts of APRICOT are:

(1) with domain-dependent knowledge, especially deep knowledge (such as knowledge of description of the system and devices) and commonsense knowledge (such as physical laws), and with constraints, hypotheses are generated or enumerated automatically, and

(2) the maintenance of multiple contexts is regarded as a mechanism of inference control and is considered as a general procedure providing an interface between a domain-independent truth maintenance system and a domain-dependent problem solver.

4.1 APRICOT Architecture

APRICOT provides a basic framework for using hypothetical reasoning in problem solving, and performs the management of multiple contexts [2] on the basis of a logic-based TMS. APRICOT itself consists of an inference engine, knowledge bases and modules for the *generation*, *selection* and *verification* of hypotheses, to enable hypothetical reasoning. Figure 1 is a block diagram of the APRICOT system configuration.

[2] A combination of hypotheses is called an *environment*, and the set of all data which hold in an environment is called a *context* [de Kleer 86a].

Knowledge bases contain domain-dependent knowledge such as deep knowledge, commonsense knowledge, problem solving knowledge and heuristic knowledge. Some of these items of knowledge may be given as constraints. An *inference engine* consists of a domain-dependent problem solver and a general or domain-specific scheduler. A *problem solver* performs various kinds of inference depending on the problem domain, and in hypothetical reasoning modules, the TMS acts as a domain-independent management mechanism to manage dynamic knowledge and multiple contexts. There are also some general problem solving strategies performed by a general *scheduler* which acts as the interface between the TMS and the problem solver; one of these is *dependency-directed search* (DDS). DDS is a search mechanism based on *dependency-directed backtracking* (DDB), with intelligent caches [Stallman & Sussman 77] to avoid redundant computing and rediscovering failures involved in chronological backtracking. DDS itself plays an important role in TMSs such as the management of dynamic knowledge (preservation of a contradiction-free state) and guidance for problem solving.

The role of each hypothetical reasoning module is as follows, and is shown in Figure 1.

1. **Hypothesis generation** (and/or **enumeration**): This module determines the *hypothesis space* from the knowledge bases, and from them generates (or enumerates) a *set of hypotheses* consisting of meaningful elements for problem solving. This module may be started up by invoking hypothetical reasoning from the domain-dependent problem solver. Depending on the problem, domain-dependent knowledge and deep knowledge (such as knowledge of descriptions of the system and devices) may be used to enumerate a set of feasible hypotheses automatically. Hypothetical generation may need advanced reasoning mechanisms such as induction and analogy. Hypothesis generation (or enumeration) may take place all at once, or in stages, or hierarchically. Depending on the circumstances of the problem, hypothesis generation may be omitted. with hypotheses given explicitly by the user

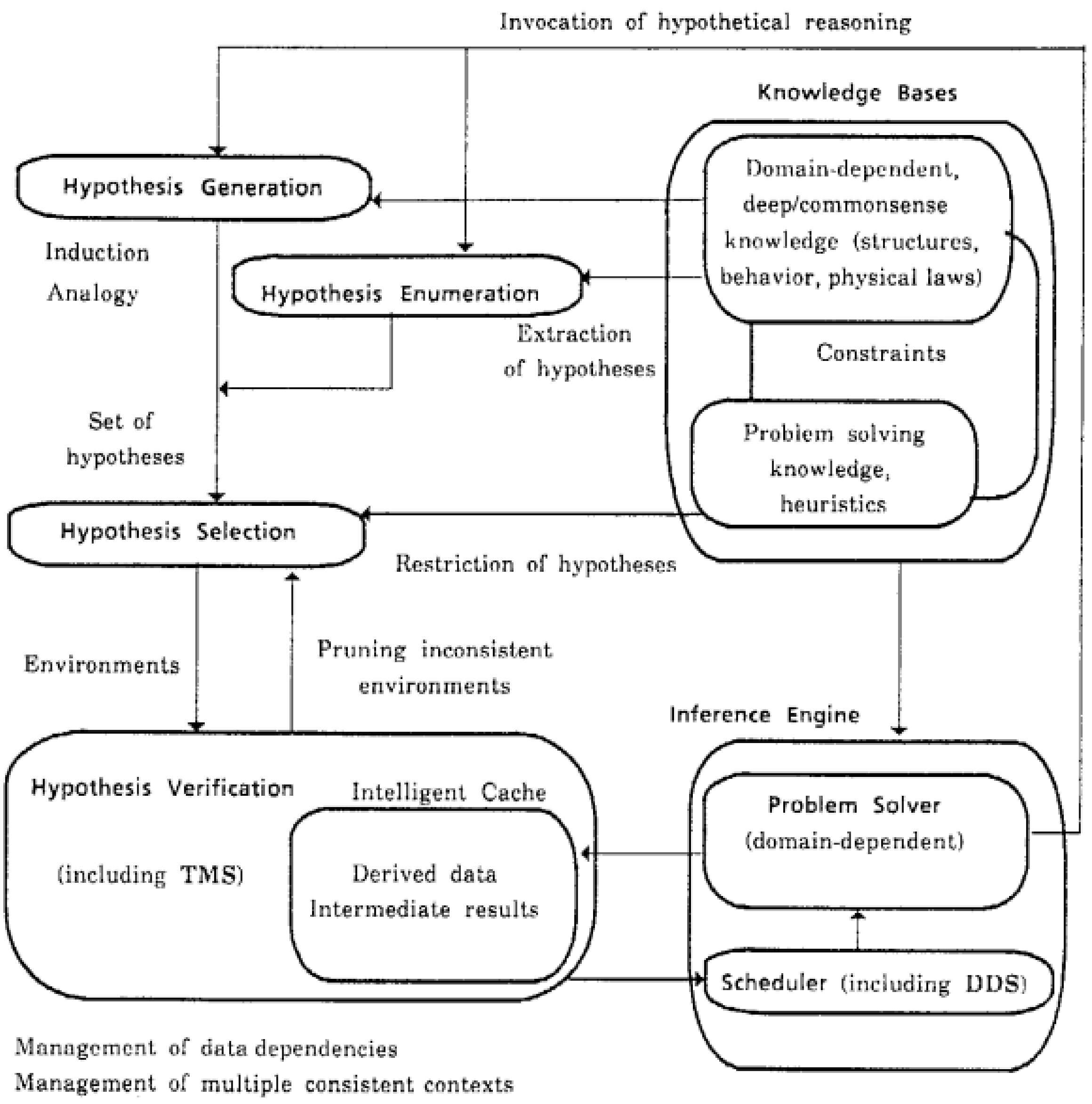
Invocation of hypothetical reasoning
Knowledge Bases
Hypothesis Generation
Domain-dependent, deep/commonsense knowledge (structures, behavior, physical laws)
Induction
Analogy
Hypothesis Enumeration
Extraction of hypotheses
Constraints
Set of hypotheses
Problem solving knowledge, heuristics
Hypothesis Selection
Restriction of hypotheses
Environments
Pruning inconsistent environments
Inference Engine
Hypothesis Verification
Intelligent Cache
Problem Solver (domain-dependent)
(including TMS)
Derived data
Intermediate results
Scheduler (including DDS)
Management of data dependencies
Management of multiple consistent contexts

Figure 1. APRICOT architecture

or by the problem solver instead.

2. **Hypothesis selection:** From the set of hypotheses provided by the hypothesis generation module, those elements which are candidates for acceptance as *hypotheses* are selected, and *environments* which are combinations of those hypotheses creating their *contexts* are submitted to be considered to the hypothesis verification module. The results of the hypothesis verification module are fed back to the hypothesis selection module to avoid the selection of inconsistent or unnecessary

hypotheses. By incorporation with problem solving knowledge or several kinds of heuristics, it restricts environments and determines their search order. Together with hypothesis verification, this constitutes the control of searches through an order of hypothesis expansion. This is sometimes reduced by giving the partial order of hypotheses explicitly.

3. **Hypothesis verification:** Reasoning is conducted based on the hypotheses selected by the hypothesis selection module, and the *validity* of the hypotheses is judged by using the assertions, which are supplied by the problem solver, in a database called the *intelligent cache.* In this process, a *truth maintenance* mechanism capable of handling multiple contexts is used to manage the dependency relations between data and assertions in the databases. When a contradiction is discovered, the environments directly related to the failure are rejected, thereby exercising control over the hypothesis selection module. This module manages multiple consistent contexts like the ATMS, and through the scheduler, the focus is determined to control the problem solver.

4.2 Proposals to Solve Problems

1. *Domain-dependent problem solvers are needed:*

In APRICOT, we are forced to consider system configurations for separate types of problems employing appropriate approaches, bearing in mind the types of hypotheses which can be handled, and narrowing down the types of inference functions to be used. In particular, many aspects of the hypothesis generation and verification processes must be considered in the light of the application. For instance, sets of feasible hypotheses may be generated using domain dependent knowledge and deep knowledge.

2. *Multiple extension problems are unsolved:*

We shall regard the maintenance of multiple contexts as a mechanism of inference control, and consider DDS as a general procedure providing an interface between the TMS and the problem solver. By focusing on the logic behind the search procedure, we will attempt an approach in which search control is via an AND/OR tree, which is more natural to describe a problem [Inoue 88a]. [3] Because of this, not every hypotheses will be handled concurrently, as is the case with the ATMS; and frameworks may be employed which allow for the incremental addition of hypotheses along their contexts, and improve the search efficiency more than the scheduling mechanism for the ATMS [de Kleer 86b] or *assumption-based DDB* [de Kleer & Williams 86].

3. *Other problems:*

(1) - (5): These problems are strongly domain-dependent. As an approach to practical application, the aspects of application problems which hypothetical reasoning is most effective in dealing with must be considered, and systems designed accordingly.

(6) and (7): The general scheduler has several advantages as it incrementally searches an AND/OR tree with a hierarchical structure [Inoue 88a]. Especially, problem solving can proceed efficiently with *compiled knowledge* because a contradiction in an intermediate level can be found so that a kind of compilation of some condition on a set of low level knowledge can be represented.

5. Application to Design

Domain-independent hypothetical reasoning modules and context searches by the general scheduler must be used as appropriate, in tandem with reasoning strategies corresponding to the problem domain under consideration. Design and planning con-

[3] The resulting algorithm is like AO* or GBF [Pearl 84], where different solutions are identified by a *solution tree*, each one representing a possible context. When some estimation or preference can be obtained among assumptions or environments, one or all optimal solutions can be obtained. For details, see [Inoue 88a].

stitute one conceivable application area. *Generate & Test* (G&T) can be contained within frameworks for hypothetical reasoning, as mechanisms for the generation, selection and verification of hypotheses. There are many attractive research areas such as the interrelations between hypothetical reasoning on the one hand, and the handling of constraints, redesign, trial-and-error processes, failure recovery, and other matters, on the other.

Of these, the relation between hypothetical reasoning and constraints may be regarded as essentially a *constraint satisfaction problem* (CSP). In CSPs, consistent assignment of values for a set of variables which satisfies all constraints is to be found [Mackworth 77]. A constraint, $C_i(X_{i_1}, \ldots, X_{i_j})$, specifies which values of the variables are compatible with each other. CSPs can in essence be solved by a G&T approach; the constraints are used to test consistency of the assignments made by a generator. When the constraints can be applied to partial assignments, partial solutions can be pruned by hierarchical problem solving (called *hierarchical G & T*), which is more efficient. By applying backtracking in search control, the causes of failures can be analyzed and retained in memory, for use as guides in subsequent processing, so that similar failures do not occur again. Here, the truth maintenance approach can be applied profitably. Chronological backtracking is used to implement hierarchical G&T, but DDS can enhance performance more.

The APRICOT system can be applied to CSPs as follows. In CSPs, hypotheses are regarded as assignments of values to variables. When an observation is discovered in a particular context, hypotheses supporting that observation or its negation are searched for, and any occurrence of contradictions within the context is checked. With the constraints folded into the generator, the problem solver can assign values for variables like *constraint propagation*. Hierarchical G&T with DDS makes the search more efficient. An AND/OR tree search in APRICOT is then applied to the specification of the problem. At any level of the tree, when a hypothesis is newly considered, the TMS can check whether it is consistent with a current environment by computing supporting hypotheses of it or its negation. It never generates an environment that occurs in an impossible combination. We emphasize that the scheduler can treat problems with a hierarchical structure. A hypothesis in an intermediate level can represent compiled knowledge, so that a partial solution tree can be pruned by the constraints if inconsistencies are found in an intermediate level. We can also represent 'components', that is, partial solutions to a set of constraints which can then be ignored because the constraints are already implicit in themselves. [4]

It should be also noted that this method can be applied to a type of design task other than simple CSPs, such that the problems need to be selected for their design models as well as for the values of the variables for models. A *design model* can be represented by a set of constraints relating the desire, intention, specification or necessity given by the user forming a context. Such models can be represented by the hierarchical structure of AND/OR graphs, and the parameters of the models can be attached below them. The difficulty is in handling dynamic constraints. The following two cases are conceivable:

1. *Relaxation of weak constraints*, and
2. *Constraints created during a problem solving process.*

One way of dealing with such problems could be to construct hypothesis worlds (design models), in which constraints are either valid or invalid. The assumption-based approach is very helpful for selecting models, as it maintains multiple contexts elegantly.

By addressing such problems in applications, it will be possible to construct the practical hypothetical reasoning system, which will serve as the core for controlling various systems designed to solve specific problems. In ICOT, mechanical design, especially

[4] This method of representation is reported to be very useful for CSPs in independent research by [Mittal & Frayman 87].

routine design for the power transmission mechanism for a lathe [Inoue *et al.* 88], has been selected as an example, so that hypothetical reasoning is utilized with knowledge compilation and constraint logic programming.

6. Conclusion

This paper discussed an architecture and techniques for problem solving that utilize hypothetical reasoning. Especially, the general search procedure in APRICOT works between the domain-dependent problem solver and the domain-independent TMS which maintain dynamic knowledge. The main characteristics of the proposed method are that reasoning is controlled by an AND/OR tree search mechanism, and that hypotheses can be added to the TMS along their contexts incrementally rather than added to every possible world concurrently in a flat structure like the ATMS. This mechanism can solve constraint satisfaction problems in design elegantly.

The proposed framework will be incorporated into a hypothetical reasoning system, APRICOT, to be developed. We have already developed a system implemented in ESP based on the ATMS and a general problem solver [Iijima & Inoue 88], and are working on a sophisticated scheduler and a constraint compiler to be attached to them. APRICOT will supply the basic architecture for maintaining multiple contexts based on a logic-based TMS, and will be the core for controlling various reasoning in a new-generation knowledge system tool. We shall apply this mechanism to constraint solving in mechanical design.

Acknowledgments

I would like to thank Dr. Koichi Furukawa, Mr. Yuichi Fujii and Dr. Ryuzo Hasegawa of ICOT for their useful comments and encouragement. The discussion with Professor Mitsuru Ishizuka of the University of Tokyo, Professor Hiroshi Nakagawa of Yokohama National University and the other members of the ICOT KSS-Working Group HYR-SWG were very helpful in refining the ideas described in this paper. I also wish to express my thanks to Dr. Kazuhiro Fuchi, Director of ICOT Research Center, who provided me with the opportunity to pursue this research in the Fifth Generation Computer Systems Project.

References

[Bobrow 80] Bobrow, D. G. (ed.), *Special Issue on Nonmonotonic Logic, Artificial Intelligence* **13** (1980), pp.1–174.

[de Kleer 86a] de Kleer, J., "An Assumption-based TMS", *Artificial Intelligence* **28** (1986), pp.127–162.

[de Kleer 86b] de Kleer, J., "Problem Solving with the ATMS", *Artificial Intelligence* **28** (1986), pp.197–224.

[de Kleer & Williams 86] de Kleer, J. and Williams, B. C., "Back to Backtracking: Controlling the ATMS", *Proc. AAAI-86* (1986), pp.910–917.

[Doyle 79] Doyle, J., "A Truth Maintenance System", *Artificial Intelligence* **12** (1986), pp.231–272.

[Finger & Genesereth 85] Finger, J.J. and Genesereth, M.R., *RESIDUE: A Deductive Approach to Design Synthesis*, Technical Report STAN-CS-85-1035, Department of Computer Science, Stanford University, 1985.

[Hanks & McDermott 87] Hanks, S. and McDermott, D., "Nonmonotonic Logic & Temporal Projection", *Artificial Intelligence* **33** (1987), pp.379–412.

[Iijima & Inoue 88] Iijima, K. and Inoue, K., *An ATMS implemented in ESP (Ver. 1)*, ICOT Technical Memorandum TM-467, ICOT, 1988 (in Japanese).

[Inoue 88a] Inoue, K., "Pruning Search Trees in Assumption-based Reasoning", *Proc. Avignon '88: The 8th International Workshop on Expert Systems & their Applications* (1988).

[Inoue 88b] Inoue, K., *On the Semantics of Hypothetical Reasoning and Truth Maintenance*, ICOT Technical Report TR-356, ICOT, 1988.

[Inoue *et al.* 88] Inoue, K., Nagai, Y., Fujii, Y., Imamura, S. and Kojima. T., *Analysis of the Design Process of Machine Tools — Example of a Machine Unit for Lathes*, ICOT Technical Memorandum TM-494, ICOT, 1988 (in Japanese).

[Mackworth 77] Mackworth, A. K., "Consistency in Networks of Relations", *Artificial Intelligence* **8** (1977), pp.99–118.

[McDermott 83] McDermott, D., "Contexts and Data Dependencies: A Synthesis", *IEEE Trans. Pattern Anal. Machine Intelligence* **5** (3) (1983), pp.237–246.

[Mittal & Frayman 87] Mittal, S. and Frayman F., "Making Partial Choices in Constraint Reasoning Problems", *Proc. AAAI-87* (1987), pp.631–636.

[Pearl 84] Pearl, J., *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, Reading, MA, 1984.

[Poole 86] Poole, D. L., *Default Reasoning and Diagnosis as Theory Formation*, Technical Report CS-86-08, Department of Computer Science, University of Waterloo, 1986.

[Reiter & de Kleer 87] Reiter, R. and de Kleer, J., "Foundations of Assumption-based Truth Maintenance Systems: Preliminary Report", *Proc. AAAI-87* (1987), pp.183–188.

[Stallman & Sussman 77] Stallman, R. M. and Sussman, G. J., "Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis", *Artificial Intelligence* **9** (1977), pp.135–196.