Blackboard Scheduler Control Knowledge for Recursive Heuristic Classification

Young-Tack Park*

ABSTRACTS

Dynamic and explicit ordering of strategies is a key process in modeling knowledge-level problem-solving behavior. This paper addressed the important problem of how to make the scheduler more knowledge intensive in a way that facilitates the acquisition, integration, and maintenance of the scheduler control knowledge. The solution approach described in this paper involved formulating the scheduler task as a heuristic classification problem, and then implementing it as a classification expert system. By doing this, the wide spectrum of known methods of acquiring, refining, and maintaining the knowledge of a classification expert system are applicable to the scheduler control knowledge.

One important innovation of this research is that of recursive heuristic classification : this paper demonstrates that it is possible to formulate and solve a key subcomponent of heuristic classification as heuristic classification problem. Another key innovationn is the creation of a method of dynamic heuristic classification : the classification alternatives that are selected among are dynamically generated in real-time and then evidence is gathered for and against these alternatives. In contrast, the normal model of heuristic classification is that of structured selection between a set of preenumerated fixed alternatives.

I. INTRODUCTION

Modeling problem-solving is carried out at a knowledge level and is considered as constructing abstract strategy knowledge (Clancey, 1987) (Wielinga et al., 1991). Over many years, several task-specific shells have been developed for a class of generic problems in different domains, for example, heuristic classification (Clancey, 1987),cover-

본 연구는 1995년 숭실대학교 교내학술 연구비 지원에 의하여 연구되었음.

* School of Computing Soong Sil University Seoul, Korea

and-differentiate (Eshelman and McDermott, 1987), propose-and-revise (Marcus, 1988), and generic tasks (McDermott, 1988) (Chandrasekaran, 1989). A basic motivation behind these researches is that strategy knowledge should be more explicit to facilitate diverse dimensions of expertise (Karp and Wilkins, 1989) (Clancey, 1992). From this perspective, several frameworks are proposed for modeling the required expertise for different tasks (Clancey, 1987) (Wielinga et al., 1991).

Such generic knowledge can be used as standards for building a class of knowledge-based systems. However, inspite of the motivation of these systems to represent control knowledge explicitly, their methods put limitations to represent more explicit control knowledge. For instance, orders of tasks and meta-rules are hard-wired by human designers and the rationale behind this ordering becomes implicit. We do believe however that strategy knowledge should be more explicit to represent intentions behind orders of tasks and meta-rules.

We propose a four-layered framework to represent control knowledge for heuristic classification. Each layer represents its own functional capability in problem-solving. The framework consists of inference engine, scheduler control knowledge, strategy control knowledge, and domain knowledge. In addition, it uses a separate scheduler which employs a scheduler blackboard and scheduler control knowledge to describe alternative inference processes and to evaluate utilities of them in a more explicit way.

One important innovation of this research is that of recursive heuristic classification : this paper demonstrates that it is possible to formulate and solve a key subcomponent of heuristic classification as a heuristic classification problem. Another key innovation is the creation of a method of dynamic heuristic classification : the classification alternatives that are selected among are dynamically generated in real-time and then evidence is gathered for and against these alternatives. In contrast, the normal model of heuristic classification is that of structured selection between a set of preenumerated fixed alternatives.

II. MODELING EXPERTISE

Several formalisms are used to represent generic strategy knowledge, such as metarules (Davis, 1980), blackboard control (Hayes-Roth, 1985) (Engelmore and Morgan, 1988), tasks and metarules (Clancey, 1987), procedural formalism (Cohen et al., 1987a), and strategy rules (Cohen et al., 1987b) (Gruber, 1989). However, they hide much of the knowledge underlying order of strategies which are represented by tasks, metarules, or knowledge sources. Our intent is to distinguish control knowledge which determines orders of strategic actions from control knowledge which envisions applicable strategic actions. A basic premise underlying our approach is to represent control knowledge more explicitly. From this perspective, we decompose control knowledge into scheduler control knowledge and strategy control knowledge. While strategy control knowledge specifies what strategic action to do to achieve a particular goal, scheduler control knowledge specifies why a particular strategic action is more preferable.

This scheduling control knowledge must determine utilities of strategic actions from the perspective of current problem-solving states, examine contents of strategy rules that suggest the strategic action, and see if the action is coherent with previous actions to achieve a goal.

Scheduler control knowledge is essential for multiple uses. First of all, this knowledge enables problem-solvers to determine an appropriate action dynamically. Unlike previous hard-wired control of ordering tasks and meta-rules, this knowledge can be used to take next actions opportunistically. Second, explanation systems can access explicit scheduling process which is described by scheduler control knowledge. Hence, deeper explanations can be provided to explain why a strategic action is taken and is preferred at a point. This allows external agents, such as users, knowledge engineers, and learning programs, to understand how systems solve a problem. Third, scheduler control knowledge can be used for differential modeling. Differential modeling (Sleeman and Brown, 1982) (Wenger, 1987) is a method to model external agent's problem-solving behavior in terms of knowledge structures of expert systems. This method is essential in apprenticeship-based programs (Wilkins, 1988) which watch external agent's problem-solving behavior. From the experience of ODYSSEUS (Wilkins, 1988), we believe a challenging problem in differential modeling is to model diverse problem-solving behavior of experts. In order to model such diverse behavior, scheduler control knowledge is necessary to contrast between applicable actions with explicit knowledge.

2-1. Strategy Control Knowledge Source Chains

To facilitate opportunistic problem-solving and differential modeling, strategy control knowledge is represented by a set of strategy control knowledge sources. A distinguishing feature of strategy knowledge sources lies in transparent representation. Premise clauses of strategy knowledge sources are declarative and can be accessed by other programs on the fly. This allows scheduler control knowledge to examine contents of strategy knowledge sources to evaluate strategic actions dynamically. In addition, strategy knowledge sources are represented by abstract terms and can be driven backwards (Wilkins, 1988) as well as forwards. Declarative representation allows strategy knowledge sources to be driven backwards, from an observed action to plausible higher goals, as well as forwards, from a goal to an action. This meta-rule backchain method (Wilkins, 1990) is essential to explain an observed action in apprenticeship learning and tutoring.

Examples of strategy control knowledge sources are shown in Fig 1. Note that strategy control knowledge sources are represented by abstract terms, such as ***differential*** and ***evidence-for***. Arguments of the abstract terms are unified with domain-specific facts on the fly. For instance, *H1* and *H2* may be bound to ***meningitis*** and ***migraine***. When a strategy knowledge source is activated by a change on the blackboard and its premise clauses are satisfied, it invokes subgoals. As is evident in Fig 1, ***differentiate-hyp*** strategy knowledge source invokes ***applyrule*** knowledge source, which in turn activates ***findout*** strategy knowledge source. An example of such a knowledge source chain on the fly can be ***goal(differentiate-hyp(meningitis, migraine))***$\leftrightarrow$***goal(applyrule(rule1010))***

```
goal(differentiate-hyp(H1, H2)) :                                  goal(findout(F1)) :
differential(H1, H2)                                               subsumes(F1, F2),
suggested-by(H1, H2, F),                                           not concluded(F1),
evid-for(F, H1, R1, B1),          goal(applyrule(R)) :             boolean(F2),
evid-for(F, H2, R2, B2),          not rule-applied(R),             not concluded(F2),
evid-conflicting(B1, B2),         has-premise(R, F1),              redflag(F2),
subgoal(applyrule(R1)),           unknown(F1),                     askfirst(F2),
subgoal(applyrule(R2)             subgoal(findout(F1)).            goal(findout(F2)).
```

Figure 1. Strategy Control Knowledge Sources

↔*findout(focalsigns)*↔*findout(diplopia)*. This knowledge source chain shows why *findout(diplopia)* action is applicable. Besides, the scheduler can access premise clauses of strategy knowledge sources in the chain. Hence, the scheduler can see *subsumption* relation between *focalsigns* and *diplopia* and *diplopia* is a *redflag* finding by examining transparent premise clauses in *findout* knowledge source. This capability to examine contents of knowledge source chains enables the scheduler to evaluate the utility of a strategic action. For instance, the fact that an action is invoked by a knowledge source which has a *redflag* premise clause means that the action may be useful to diagnose a case at the point.

2-2. Recursive Heuristic Classification

The MINERVA expert system shell represents strategies of a generic task, heuristic classification (Clancey, 1985). Heuristic classification is the process of selecting a solution out of a pre-enumerated solution set using heuristic knowledge. In this paper, the MINERVA shell was extended by the addition of *recursive heuristic classification*. We view the ordering problem of strategic meta-rules as a separate problem-solving task and formulate and solve this scheduling task of heuristic classification as a heuristic classification problem.

In MINERVA, the strategic scheduler selects a strategy control knowledge source chain from a set of dynamically generated knowledge source chains using heuristic knowledge. Just as heuristic classification selects a solution out of a set of pre-enumerated solutions, the strategic scheduler selects a strategy control knowledge source chain from a set of knowledge source chains. The scheduling subtask of heuristic classification is solved by the heuristic classification method in a recursive way. MINERVA-medicine uses strategic control knowledge sources to generate a set of knowledge source chains (KSC) dynamically and MINERVA-scheduler uses the same heuristic classification strategic knowledge to select a desirable knowledge source chain.

The recursive heuristic classification differs from heuristic classification in defining a set of pre-enumerated solutions. While heuristic classification inputs a fixed set of solutions, recursive heuristic classification handles a set of *dynamically generated knowledge source chains*. Knowledge source chains are generated on the fly and are handled by the scheduler. Hence, in different scheduling sessions, this set contains different solutions to be handled by recursive heuristic classification.

From the observation that heuristic classification can be a reasoning method for the scheduling subtask, it is easy to anticipate that the same strategy knowledge can be used for MINERVA-medicine and MINERVA-scheduler systems. Strategy knowledge in MINERVA-medicine accesses application domain knowledge, such as medicine and circuit knowledge, to collect diagnostic evidences and hypotheses. Similarly, the same strategy knowledge is used in MINERVA-scheduler to collect scheduling evidences for and against dynamically generated knowledge source chains using scheduler control knowledge. Since a problem-solving step in MINERVA-medicine is important for the performance and the explanation, the scheduler is replaced by a separate knowledge-based system, MINERVA-scheduler. On the other hand, in MINERVA-scheduler, the final scheduling result is only interested. Its intermediate scheduling processes are considered as less important steps. Hence, MINERVA-scheduler uses a simple first-in-first-out(FIFO) scheduler for scheduling knowledge-source chains in MINERVA-scheduler. MINERVA consists of domain and scheduling blackboards to facilitate opportunistic problem-solving and robust differential modeling. In problem-solving, MINERVA-medicine generates a set of applicable knowledge source chains on the scheduling blackboard, and MINERVA-scheduler examines these strategy knowledge source chains to select the most appropriate one dynamically. In differential modeling, the rational behind ordering of strategic actions are explicitly described on the scheduling blackboard and can be accessed by a differential modeler.

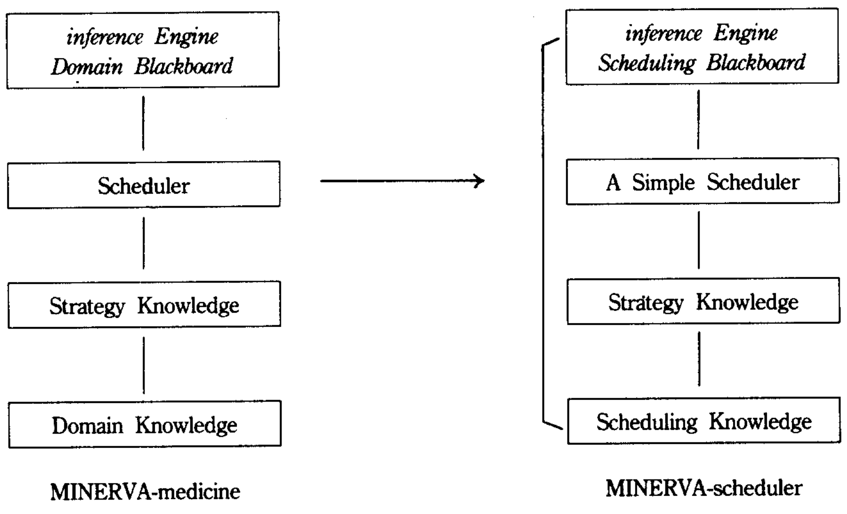

Figure 2. A MINERVA-based scheduler

III. Creation of Scheduler Control Knowledge

Scheduler knowledge consists of scheduler facts and scheduler rules. A set of scheduling vocabulary is used to represent scheduling evidences from dynamic states and knowledge source chains. In addition, for a focused exploration of scheduling facts, hierarchies between scheduling facts are defined. Heuristic classification strategy knowledge uses these hierarchies to collect scheduling evidences. Scheduling rules use scheduling evidences to evaluate and differentiate between knowledge source chains generated by MINERVA-medicine.

3-1. Define Vocabulary

A set of vocabularies is developed to characterize problem-solving states and knowledge source chains of MINERVA-medicine. The scheduler uses three kinds of vocabularies to characterize problem-solving states of MINERVA-medicine : static information, dynamic information, and characteristic scheduling features of knowledge source chains and problem-solving states.

Vocabularies for static information come from abstract terms used for modeling domain models of MINERVA-medicine. Examples of these vocabularies are finding(X), hypothesis(Y), subsumes(A, B), etc. During a problem-solving session, each variable is bound to a domain term. For instance, finding(X) may be instantiated as finding (headache), finding(fever), etc. Hypothesis(Y) may become hypothesis(meningitis), hypothesis(migraine), etc. MINE-RVA-scheduler also needs a set of vocabulary to extract useful scheduling evidences from dynamic problem-solving states of MINERVA-medicine, such as differential(X), rule-fired(Y), finding-known(Z), etc.

Based on these vocabularies, two kinds of vocabularies for characteristic scheduling features is developed : the state vocabulary and the knowledge source chain vocabulary. The state vocabulary represents abstract states of MINERVA-medicine's problem-solving states. Examples of the state and the knowledge source chain vocabulary are shown in Figure 3-(a). Since MINERVA-medicine is a blackboard based system, its problem-solving state can be represented at each blackboard cycle. MINERVA-scheduler reasons about the states of MINERVA-medicine at particular time intervals. As shown in Fig 3-(a), MINERVA-scheduler abstracts problem-solving states of MINE-RVA-medicine at each problem-solving cycle.

Fig 3-(b) shows examples of knowledge source chain vocabularies which examine transparent knowledge source chains. MINERVA-scheduler examines MINERVA-medicine's knowledge source chains to find out characteristic features which are useful for scheduling these knowledge source chains. For instance, a knowledge source chain which differentiates active hypotheses or clarifies an important finding deserves to be considered before other knowledge source chains. Transparent representation of strategy knowledge source chains allows the scheduler to examine each applicable knowledge source chains and uses these vocabularies to describe characteristic features of them.

no-change-of-focus(t_{i-1}, t_i)

Focus of MINERVA-medicine has not been changed between t_{i-1} and t_i cycles.

differential-non-null(t_i)

At t_i cycle, MINERVA-medicine has no active hypotheses.

(a) State vocabularies

chain-contains(*KSC*$_i$, t_j, *strategy-predicate*(*redflag*))

At t_j cycle, *KSC*$_i$ (knowledge source chain) contains a redflag predicate.

chain-contains(*KSC*$_i$, t_j, *strategy*(*diff-hypotheses*))

At t_j cycle, *KSC*$_i$ is activated by a differentiate hypotheses strategy.

(b) Knowledge source chain vocabularies

Figure 3. Examples of characteristic features

differential-non-null(t_i) $\overset{\text{subsumes}}{\rightarrow}$ size-of-differential-large(t_i)

differential-non-null(t_i) $\overset{\text{subsumes}}{\rightarrow}$ size-of-differential-large(t_i)

differential-non-null(t_i) $\overset{\text{subsumes}}{\rightarrow}$ size-of-differential-large(t_i)

(a) States of active hypotheses

hard-data-action-exists(t_i, has-goal(*KSCm*, Goal)) $\overset{\text{parent}}{\rightarrow}$

soft-data-action-exists-that-support-the-same-goal(t_i, has-goal(*KSCn*, Goal)) $\overset{\text{parent}}{\rightarrow}$

hard-data-action-gives-more-supports-to-goal(t_i, *KSCm*, *KSCn*, *Goal*))

(b) Characteristics of knowledge source chains

Figure 4. Examples of hierarchy trees

3-2. Define Hierarchy

In order to find characteristic features from problem-solving states and knowledge source chains of MINERVA-medicine efficiently, MINERVA-scheduler defines subsumption relations between characteristic features. Basically, scheduler facts are organized into a set of hierarchy trees, each of which has a root. These roots are classified as initial-data of MINERVA-scheduler. Heuristic classification strategy knowledge finds out these scheduling initial data and continues to explore hierarchy trees whose root's values are true. Fig 4 shows examples of hierarchy trees.

Fig 4-(a) shows a hierarchy tree which represents the sates of active hypotheses of MINERVA-medicine.

It has a root, differential-non-null(t_i), and has three children. During the scheduling phase, the scheduler tries to find out values of three children only when a scheduling initial data, differential-non-null(t_i), becomes true. Fig 4-(b) shows a taxonomy tree which is used to represent features of knowledge source chains. This tree describes a particular state where there are two knowledge source chains which attempt to achieve the same goal. Suppose a knowledge source chain suggests to find out a soft finding to achieve the goal and another suggests to find out a hard datum. In general, because a hard datum is expensive to acquire the value, a knowledge source chain with a soft finding is preferred to another with a hard datum. However, if the knowledge source chain with the hard datum strongly supports the goal, it is plausible to select this knowledge source chain. To model this kind of scheduling behavior, the scheduler needs to examine the contents of knowledge source chains. And, the transparent knowledge source chain in MINERVA allows this deep reasoning to schedule knowledge source chains.

```
if    knoledge-souce-chain(KSCi),
      chain-contains(KSC_i, t_k, strategy(clarify-finding)),
      chain-contains(KSC_i, t_k, focus(Focus)),
      member(Focus, chief-complaints),
      not chief-complaints-clarified(t_k)
      knowledge-souce-chain(KSC_j)
      chain-contains(KSC_j, t_k, strategy(process-finding)),
      chain-contains(KSCj, t_k, focus(Focus)),
then  conclude(KSC_i, 500), conclude(KSC_j, −100),
```

If a knowledge source chain suggest to clarify a chief complaint and another infers hypotheses, then the former is preferred (.5) and the latter is not preferred (−.3).

Figure 5. Task ordering scheduler knowledge

3-3. Define Inference Rules

Scheduler inference rules evaluate and contrast between knowledge source chains generated by MINERVA-medicine at a given problem-solving state. The scheduler finds characteristic features of both the problem-solving state and the knowledge source chains of MINERVA-medicine, then uses inference rules to evaluate applicable knowledge source chains of MINERVA-medicine. A scheduler inference rule is represented by a production rule, that is if-then types of implication formulas. The if side specifies a set of conditions of abstract features for the rule's applicability. The then side symbolizes the implications to be drawn with a probability.

There are three types of scheduler rules. First, some scheduler rules evaluate applicable strategy knowledge source chains according to a problem-solving state. For instance, if there are multiple strongest hypotheses on the differential whose beliefs are larger than 800, then knowledge source chains which can differentiate them are preferred with a scheduling belief of 0.9. Second, some scheduler rules represent generic task ordering knowledge. As shown in Fig 5, when chief complaints are not clarified, the scheduler prefers collecting more information to generating hypotheses. This kind of a scheduler rule is used to contrast between applicable knowledge source chains dynamically. Another scheduling rule prefers generating hypotheses to collecting information when enough information has been collected. Since ordering between tasks without considering problem-solving states tends to be strict, MINERVA uses this kind scheduling rules to order several applicable tasks dynamically. And, they intend to represent implicit rationale behind the task order explicitly. Third, some scheduler domain rules evaluate applicable knowledge source chains by reasoning utilities of actions associated with knowledge source chains. Suppose there are two feasible actions which try to achieve a same goal. One suggests to find out *soft datum* and another suggests to find out *hard datum*. In general, because *hard datum* is expensive to obtain, the former is preferred to the latter by an explicit scheduler control rule. Previous systems use implicit ordering of meta-rules to realize this scheduling (Clancey, 1987) (Clancey, 1992), but MINERVA uses explicit scheduler domain rules which reason about utilities and costs of strategic actions suggested by knowledge source chains. And, its scheduling process is explicitly represented on the separate scheduling blackboard, which allows other programs, such as explanation and learning programs, to access the scheduling process.

Ⅳ. Discussion

Explicit scheduler control knowledge has a number of advantages in opportunistic problem-solving, deep explanation, apprenticeship critiquing, knowledge-base debugging, and knowledge acquisition. Since the MINERVA's scheduling approach reasons about contents of strategy knowledge source chains dynamically based on problem-solving states, it allows MINERVA to solve a problem in a more focused way. And, preserving the scheduling process on the separate scheduling blackboard makes it possible to generate explanations at the scheduling level. Hence, MINERVA can explain what strategy is associated with an action, why this strategy is selected as the most appropriate at the point, and what domain terms are used to determine the action.

Explicit representation of scheduler knowledge enables machine learning programs to induce scheduler control knowledge. For instance, inductive learning (Quinlan, 1986) (Michalski, 1983) can be applied to construct scheduler control knowledge using a set of scheduling training instances, and then failure-driven learning can refine this initial scheduling knowledge base by reasoning scheduling failures of MINERVA.

Another advantage has to do with the apprenticeship systems. ODYSSEUS (Wilkins, 1990) has demonstrated that abstract and declarative strategy knowledge can be used to generate line of reasoning of an observed action. Intuitively, multiple explanations will be generated given an observed action (Park and Wilkins, 1990). Therefore, when modeling an observed agent's diagnostic strategies, it is necessary to contrast alternative explanations as well as generating them. While the ODYSSEUS' method generates strategic knowledge sources chains to explain an action, our scheduler control knowledge can be used to evaluate utilities of strategic knowledge source chains according to current problem-solving states and coherence with previous actions. Scheduler control knowledge plays an important role to prune irrelevant explanations of an observed action in apprenticeship learning or tutoring.

Explicit scheduler control knowledge enables a differential modeler to critique an observed agent's problem-solving behavior. For critiquing, MINERVA generates applicable strategic knowledge source chains, each of which suggests an plausible action, then evaluates utilities of these chains to find out the best action to take. Hence, MINERVA can explain why the best action is preferred to others. Suppose MINERVA observes an expert's action and finds that the action is a plausible but not the best. This is possible, because MINERVA-scheduler evaluates all the actions suggested by applicable knowledge source chains. In this case, MINERVA can provide the observed expert with the best action along with an explanation.

By adding a separate scheduler, the problem-solving speed of MINERVA becomes very slow. Intuitively, the speed can be improved by using a more efficient scheduler. For instance, because changes of problem-solving states between cycles are very slight, the scheduling process can be improved by monitoring changes and updating only the scheduling evidences which are affected by the changes. To evaluate costs of adding scheduler level,

Experiment parameters		MINERVA with a simple scheduler	MINERVA with a MINERVA-scheduler
Time	Deliberation time	11.9 sec	4.1 sec
	Scheduling time	0.1 sec	214.3 sec
	Execution time	1.24 sec	0.5 sec
Findings	Inferred findings	5	4
	Asked findings	19	14
	Relevant findings	4	4
Rules	Fired rules	18	8
	Relevant findings	2	2
Hypotheses	Considered hypotheses	4	3
	Active hypotheses	2	1
Problem-solving cycles		39	24

Figure 6. Cost of adding scheduler control level

we experimented using a collection of 112 solved medical cases which were obtained from records at Stanford Medical Hospital. For this purpose, two MINERVA-based systems were constructed : a MINERVA-based system with a simple scheduler and a MINERVA-based system with a MINERVA-scheduler. Fig 6 shows experimental results concerning the effect of adding scheduler control level. As expected, average scheduling time to solve a medical diagnostic problem increases from 0.1 sec to 214.3 sec. However, average numbers of problem-solving cycles of MINERVA-medicine, considered findings, fired rules, considered hypotheses are decreased by adding a scheduler.

V. CONCLUSION

MINERVA demonstrates a more knowledge-intensive approach to support dynamic ordering of applicable knowledge source chains. Transparent representation of strategy knowledge allows MINERVA to examine contents of knowledge sources. Recursive and dynamic heuristic classification is created to formulate a key subcomponent of heuristic classification as a heuristic classification problem. By doing this, the wide spectrum of known methods of acquiring, refining, and maintaining the knowledge of a classification system becomes applicable to scheduler control knowledge.

REFERENCES

1) (Chandrasekaran, 1989) B. Chandrasekaran. Task-structures, knowledge acquisition and learning. Technical Report 89-bc-taskstr, The Ohio State Univ, 1989.

2) (Clancey 1985) W.J. Clancey, Heuristic classification, *Artificial Intelligence* 27, pp.289~350, 1985.

3) (Clancey 1987) W.J. Clancey, Acquiring, representing, and evaluating a competence model of diagnostic strategy. In *Contributions to the Nature of Expertise*, Lawerce Erlbaum Press.

4) (Clancey 1992) W.J. Clancey, Model construction operators, *Artificial Intelligence* 53, pp.1~115, 1992.

5) (Cohen et al., 1987a) P. R. Cohen, D. S. Day, J. DeLisio, M. Greenberg, R. Kjeldsen, D. Suthers, and P. Berman, Management of uncertainty in medicine, *International Journal of Approximate Reasoning* 1, pp.103~116, 1987.

6) (Cohen et al., 1987b) P. R. Cohen, D. S. Day, J. DeLisio, MU : A development environment for prospective reasoning systems, In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pp.783~788. 1987.

7) (Davis 1980) R. Davis, Meta-rules : Reasoning about control, *Artificial Intelligence* 15, pp.179~222, 1980.

8) (Engelmore and Morgan, 1988) R. Engelmore and T. Morgan. Blackboard Systems, Addison-Wesley, 1988.

9) (Eshelman and McDermott, 1987) L. Eshelman. and L. McDermott. MOLE : A knowledge acquisition tool that

uses its head, In *Proceedings of the National Conference on Artificial Intelligence*, pp.950~955, 1987.

10) (Gruber, 1989) T. Gruber. *The acquisition of strategic knowledge. Academic Press*, 1989.

11) (Hayes-Roth 1985) B. Hayes-Roth. A blackboard architecture for control, *Artificial Intelligence* 26, pp.251~321, 1985.

12) (Karp and Wilkins, 1989) P. D. Karp and D. C. Wilkins. An analysis of the distinction between deep and shallow expert systems, *Int. J. Expert Systems* 2(1), pp.1~32, 1989.

13) (Marcus 1988) S. Marcus. SALT : A knowledge-acquisition tool for propose-and-revise systems, In *Automating Knowledge Acquisition for Expert Systems*, Kluwer Academic Publishers, 1988.

14) (McDermott 1988) J. McDermott. Preliminary steps toward a taxonomy of problem-solving methods, In *Automating Knowledge Acquisition for Expert Systems, Boston* : Kluwer Academic Publishers, 1988.

15) (Michalski, 1983) R. S. Michalski. A theory and methodology of inductive inference. In *Machine Learning : an Artificial Intelligence Approach*, pages 83~134. Los Altos, CA : Morgan Kaufmann, 1983.

16) (Park and Wilkins, 1990) Y.T. Park and David Wilkins. Establishing the coherence of an explanation to improve refinement of an incomplete knowledge base. In *Proceedings of National Conference on AI, AAAI*-90, pages 511~516, Boston, MA, August 1990.

17) (Quinlan, 1986) J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1 : 81~106, 1986.

18) (Sleeman and Brown, 1982) D. Sleeman and J.S. Brown, editors. *Intelligent Tutoring Systems*. Academic Press, 1982.

19) (Wenger, 1987) E. Wenger. *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann, 1987.

20) (Wielinga et al., 1991) Wielinga, B. J., Schreiber, A., and Breuker, J. A., KADS : A modeling approach to knowledge engineering, Technical Report KADS-II/T1.1/PP/UvA/008/1.0, Univ. of Amsterdam, 1991.

21) (Wilkins 1988) D.C. Wilkins. Apprenticeship learning techniques for knowledge based systems, Technical Report STAN-CS-88-1242, Stanford Univ, CA., 1988.

22) (Wilkins, 1990) D.C. Wilkins. Knowledge base refinement as improving an incorrect and incomplete domain theory. In Machine learning : An Artificial Intelligence Apprach, Vol III. Morgan Kaufmann, 1990.