PAPER Special Issue on Intelligent CAI and Hypermedia

Student Modelling for Procedural Problem Solving

Noboru MATSUDA† *and* Toshio OKAMOTO†, *Members*

SUMMARY This study is intended to investigate a method to diagnose the student model in the domain of procedural problem solving. In this domain, the goal of an instruction should be to understand the processes of solving given problems, and to understand the reasons why problems can be solved by using sertain knowledge; the acquisition of problem solving skills might not be the intrinsic instructional goals. The tutoring systems in this domain must understand the effect of each problem solving operators, as well as when to implement these operators in order to effectively solve given problems. We have been studying and developing a system which deals with student modelling in the domain of procedural problem solving. We believe that the two types of knowledge should be clearly defined for the diagnosing tasks; *effective knowledge* (EK) and *principle knowledge* (PK). The former is the knowledge which is explicitly applied by students throughout problem solving processes, and the latter is the one which gives the justifications of the EK. We have developed a student model diagnosing system which infers students' knowledge structure pertaining to PK, based on the precedently manipulated student model about EK. This student model diagnosing method requires knowledge which argues the relationship between the PK and the EK. This knowledge plays the very important role in our system, and it's hard to describe such knowledge properly by hand. In this paper, we provide a student model diagnosing system which has the knowledge acquiring function to learn the relationship between EK and PK. The system acquires this knowledge through its own problem solving experience. Based on the student model and the acquired relational knowledge, the system can give students proper instructions about construction of EK with explanations in terms of PK. The system has been partly implemented with CESP language on a UNIX workstation.

key words: *artificial intelligence, educational systems, intelligent tutoring systems, student modelling*

1. Introduction

This study is intended to investigate a method to diagnose the student model in the domain of procedural problem solving. In this domain, students tend to learn only the problem solving skills, and essential understanding of the knowledge used to solve problems and the reasons why problems can be solved by using such knowledge is rather neglected. Suppose a student can solve a set of particular problems but fails to solve problems which are similar, this may indicate that the student does not sufficiently understand the knowledge required to solve such problems.

In general, the student model can explain the actions which are observed during students' problem solving performances [8]. Many studies on Intelligent Tutoring Systems (ITSs) are interested in predicting student's answers properly. Since such student models require quite a complicated diagnosing system, many such systems have been offered so far. The most representative student models are as follows: the bug model (e.g., [1]) which enumerates the expected errors, the perturbation model (e.g., [7]) which alters correct knowledge to be consistent with student's answers, the inductive model (e.g., [2]) which automatically generates computer programs to provide the same answers which would be responded by a student diagnosed. These diagnosing system can maintain accurate student models, but the efforts to analyze and describe domain knowledge make the task too difficult.

Suppose the system builds up an accurate student model, the correct adaption into instructional use is still being questioned. If the instructional goal is to communicate the problem solving knowledge, then identification of the knowledge which the student doesn't understand is the most important task; the diagnostic technique to recognise the student mental state isn't absolutely necessary [6].

We have been studying and developing a system which deals with student modelling in the domain of procedural problem solving [3]. This study has evolved from preceding research [5], where the problem solving knowledge for set-operations is classified into two types of knowledge; the theorem-rules which are operational rules for translating given formulae, and the conceptual knowledge of set operation which is related to the theorem-rules. We think that the two types of knowledge should be clearly defined for the diagnosing tasks; *effective knowledge* (EK) and *principle knowledge* (PK). In general, we can directly see EK applied by students throughout problem solving processes. While it is difficult to observe PK, we have developed a student model diagnosing system which infers a students' knowledge structure pertaining to PK, based on the precedently manipulated student model about EK.

The above student model diagnosing system requires knowledge which argues the relationship between

Manuscript received July 29, 1993.
Manuscript revised September 21, 1993.

†The authors are with the Graduate School of Information Systems, University of Electro-Communications, Chofu-shi, 182 Japan.

the PK and the EK. This knowledge should explain why a particular knowledge is correct, beneficial, worth memorizing, etc. So far, we have given these knowledge by hand, although the coding process is quite time consuming.

In this paper, we provide a student model diagnosing system which has the knowledge acquiring function to learn the relationship between EK and PK. The system acquires this knowledge through its own problem solving experience. Based on the student model and the acquired relational knowledge, the system can give students proper instructions about construction of EK with explanations in terms of PK.

2. Knowledge for Procedural Problem Solving

In this paper, we refer to knowledge for solving problems as problem solving knowledge. Problem solving knowledge is divided into three different classes;

(EK) **Effective Knowledge**: the knowledge called *theorems* or *formulae* which enable to solve a problem effectively.

(PK) **Principle Knowledge**: the knowledge called *axioms* or *definitions* which derive EK.

(HK) **Heuristic Knowledge**: the knowledge which implies what knowledge should be applied to the particular state.

We call the problems which require those three types of knowledge to solve, *the procedural problems*. Procedural problem solving, the subject of our investigation, is the process of searching for a solution of a given problem by applying these problem solving knowledge repeatedly.

Students, who are expected to solve given problems, have only to learn knowledge categorized as effective knowledge (EK). In general, the knowledge in EK has explicit information on how and when to apply the methods or operations. The students don't have to know the justification of each knowledge.

We believe that the knowledge categorized as principle knowledge (PK) is important to increase the understanding of problem solving. PK must explain the correctness or the origin of EK (e.g., how the knowledge is derived from PK). Therefore, students who are expected to comprehend the rationale of problem solving rather than the problem solving skills, must learn knowledge categorized as PK.

The knowledge categorized to HK, generally called heuristic knowledge, enables students to solve problems effectively. HK would choose, for example, the best operator out of many plausible candidates in a given situation. Heuristic knowledge is necessary for the students to understand PK.

3. Formalize a Domain of Procedural Problem Solving

This section formalizes the subject domain – procedural problem solving. The intent is to clarify the extent to which the student model diagnosor can work properly. Therefore, we formalize the domain of procedural problem solving from a student modelling point of view.

The model described here doesn't depend on any particular domain. We can utilize this model using many actual domains as instructional subjects. In this paper, a linear equation is our choice of a domain.

3.1 The Basic Components of the World

The followings are components which the diagnosing module mainly deals with;

- state,
- operator,
- search control rule.

The *states* represent "initial", "intermediate" and "final (goal)" states of given problems. An initial state should be given to every problem, some problems may not know the goal state at the beginning of the problem solving. The problems like "theorem proving" are given goal states explicitly, but the problems like "operation of mathematical formulae" does not know goal state (searching a goal state itself is the problem).

The *operators* suggest how to translate the state. Each operator has the conditions of applicability and the effects (i.e., transition of the *state*) caused by applying it.

Applying operators blindly at each state could cause an explosion of the search, one can never get the solution. The *search control rules* (SCRs) control the application of operators and make search efforts extremely simple.

3.2 Operator

The operators represent the operations which would decrease the distance from a particular state to the goal state. We categorize the operators into two classes;

(1) primitive operator,

(2) macro operator.

Each macro operator causes a same translation as a set of the primitive operators. Macro operators save search costs which are generated by the primitive operators of which that macro operator is consisted.

The primitive operators represent the most basic operations which can not be broken down to other operations. If the system only uses the primitive operators to solve a problem, the solution paths give students

the principled explanations in terms of domain axioms, but the searching performance is inefficient. Since the macro operators are generated from a number of primitive operators, each macro operator is explainable in terms of such primitive operators. If a one provides far more basic operations as "super primitive operators", each primitive operator is still explainable in terms of these more primitive operators. In this system the level of each operator is stated in advance to be used as a diagnostic of the student model.

Some macro operators may construct another macro operator which saves much more searching efforts. Figure 1 shows a relationship between primitive and macro operators. The macro operator "$Ax + B = Cx + D \rightarrow Ax - Cx = D - B$" consists of two macro operators; and these two macro operators consist of 2 and 3 primitive operators respectively.

3.3 Search Control Rules

The heuristic knowledge takes a very important role in the framework of general problem solving. This knowledge enhances a performance of the problem solving. If the inference engine applies matched primitive operators blindly at each state, the search space can easily exploded. The search control rules (SCRs) suggest an appropriate rule for the particular situation. The macro operator really reduced the effort of searching, then it is categorized into a kind of heuristic knowledge.

Because it is intended to use SCRs as justification of the macro operators, this system has SCRs as a set of explicitly represented knowledge. The SCRs are implemented as the knowledge to resolve the conflicts of primitive operators. That is, the SCRs give the conflicting operators their suitability in the followings three terms;

- `reject`
- `select`
- `prefer`

Those rules are called the rejection rule, the selection rule, and the preference rule, respectively.

The rejection rule implies that applying that operator would cause failure (i.e., can not fulfill the problem solving). The selection rule makes an inference engine to select the operator; that is, all other candidates fail. The preference rule implies that applying a particular operator is more suitable than others. This is realized with the quantitative evaluation which decides partially order of the probability of operators.

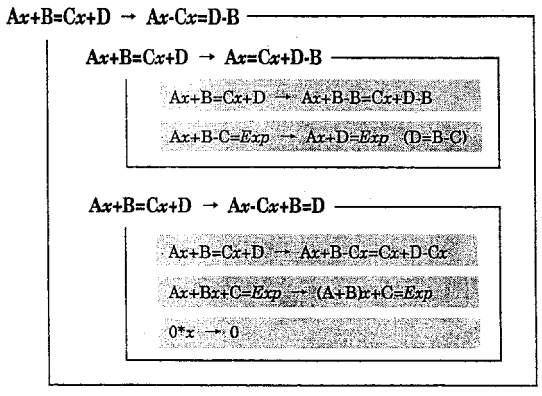
3.4 Problem Solving

The problem solving is a process of searching for a sequence of operations which translate an initial state into goal state. We call such a sequence of operators the *solution* of a problem. Figure 2 shows a process of problem solving.

The inference engine searches applicable operators at each state. All of the operators which have conditions matching the current state of working memory are equally marked applicable.

If any applicable operators more than two are brought up for any one of the states, then those competing operators are registered as the set of conflicted operators. The SCRs suggest what operator should be applied first.

After finding a solution, macro operators are generated from a sequence of primitive operators. The system can give a student the appropriate explanations of generated macro operators in terms of the SCRs and primitive operators. Figure 3 shows an example of the explanation of macro operators. The macro operator MOp_1, for example, consists of primitive operator Op_2, Op_3 and Op_4. The selection rules are applied at the state S_2 and S_3, and preference rule at S_4. These SCRs suggest that applying the operators Op_2 and Op_3 to



Bold: *Macro Rule*, Mask: *Primitive Rule*

Fig. 1 A relationship between primitive and macro operators.

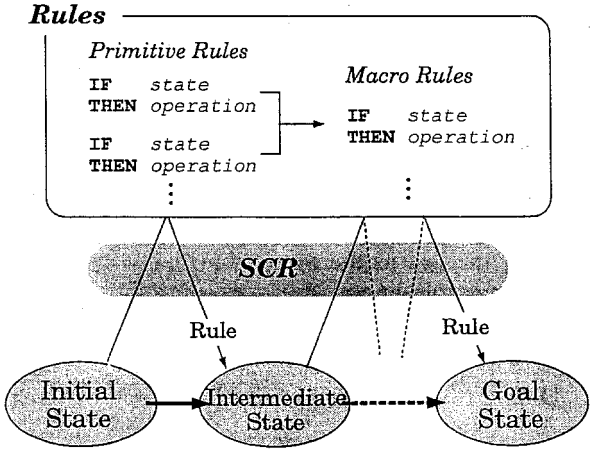


Fig. 2 The process of problem solving.

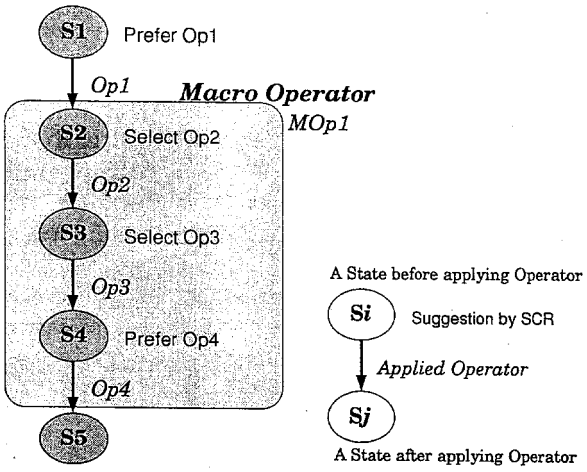


Fig. 3 The explanation of the macro operators.

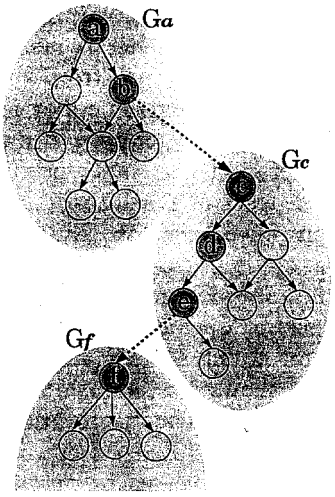


Fig. 4 An example of the diagnosis with the solution graph.

the state S_2 and S_3 is definite, and Op_4 to S_4 is preferred. The former implies all other candidates should fail, and the latter implies the operator would lead success. These notifications make a justification of how and why a macro operator MOp_1 is formed.

4. Student Model Diagnosis

This section describes the configuration of the student model diagnosor in the domain of previously formalized procedural problem solving.

4.1 The Diagnosing Issues

By observing the processes of students' problem solving, the system identifies which operators do the students know or not know. The diagnosing method traces the problem solving process with the general problem solver (GPS) [3]. The method is to infer the principle knowledge according to the comprehension of the effective knowledge; the effective knowledge is directly observable from students' performances.

There is a problem under this approach; that is, the system must know the correct relationship between effective and principle knowledge, while coding such properties is not an easy task. We have extended this technique around the following two points.

- To generalize the tracing algorithm to diagnose students' mastery state of the effective knowledge.
- To integrate automatic knowledge acquisition system which learns the relationship between effective and principle knowledge.

The following sections describe these topics.

4.2 Diagnose the Effective Knowledge

The effective knowledge is equal to the macro operators described at Sect. 3. As far as it causes no confusion, we use the term "macro operator" in place of "effective knowledge".

The system has a graph which represents the *search space* of a problem given to the students. We call this graph a *solution graph*. The system compares students' solutions with the solution graph to interpret the mastery state of their understanding of the macro operators. The system must detect the macro operators which (1) the students have understood properly, and (2) the students fail to apply to a particular situation.

Using a solution graph (SG), a student's solution path (Ps) is one of the following situations:

(1) Ps is identical with a sub-tree of SG,

(2) Ps consists of a node in SG,

(3) Ps contains a node which does not belong to any part of SG.

In situations (1) and (2), the system can easily point out the operators which the student isn't likely to know. In situation (3), the system infers macro operators which the student doesn't know by the following algorithm.

For example, suppose a student follows the correct solution path until reaching a node N_b. At this point, the student fails to apply the correct macro operator and generates a wrong node N_c from N_b. The system searches the operators that are applicable to node N_b, and marks these operators as the student doesn't seem to know. In order to follow and diagnose his or her succeeding actions, the system generates a new solution graph (G_c) with a node N_c as the initial state (i.e., root node). Figure 4 shows an example of the diagnosis of the solution graph.

4.3 Diagnose the Principle Knowledge

As described before, the macro operators consist of primitive operators. So, if a student fails to apply an

particular macro operator, it could be concluded that the student does not understand the primitive operators either. Relying on its own knowledge of the relationship between primitive and macro operators, the system infers what knowledge the student does not understand. This process is accomplished by referring to the relationship knowledge. As described in the diagnosing issues, this knowledge could be automatically acquired by the system. This knowledge is formalized by SCRs. We have developed the knowledge acquisition system based on the explanation based learning (EBL). The next section shows the configuration of the learning module.

5. Learning Search Control Rules

The system learns the search control rules (SCRs) as the knowledge of the relationship between primitive and macro operators. After solving the given problems, the learning module extracts the rules on how to control the search, and generates a database of the generic search control rules by generalizing its own problem solving experiences.

5.1 The Configuration of the Learning System

The system is given a set of SCRs which are extremely generalized, and a set of primitive operators enough to solve the general problems in the target domain.

The extremely generalized SCRs only suggest, for instance, "if an operator† has once succeeded at a node, then applying this operator to this node is preferred". As this rule is too general and is able to apply to many sates, no one can save the search efforts at all.

The track of the operators applied during problem solving is the most specialized SCR. If the system encounters the same problem, then the system applies same operator sequence as the tracked one. However, this type of SCRs can not be used for any other type of problem and it doesn't really contribute to the problem solving.

General problem solving requires appropriately specialized SCRs, which are derived from given SCRs. We use explanation based specialization (EBS) method [4] to deal with this requirement.

Figure 5 shows the configuration of the SCR learning module. It consists of (1) problem solver, (2) EBS module, and (3) domain axioms.

5.2 Problem Solver

The problem solver is a kind of the production system with the forward inference engine. It infers forward because the current domain is a mathematical equation. The problem solver selects the operator which have the conditions to be consistent with the current problem state (expression), applies the operator, and translates the problem state. This process is repeated until the

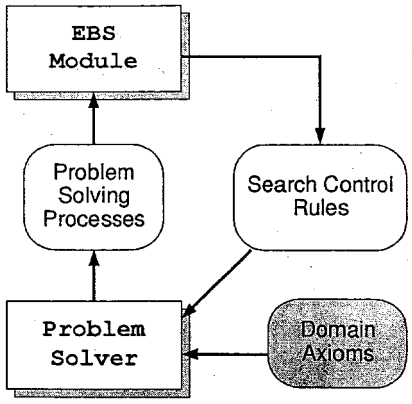


Fig. 5 The configuration of the SCR learning module.

The form of a primitive operator.

```
op( ID,
    Inquiry,
    PreConditions,
    PostConditions
  ) :-
    NextInquiry.
```

An exmaple of a primitive operator.

```
op( 0003,
    equation( e( L, R ), X ),
    [],
    [
        partially_calculable(L,Lp,Tree,Tp),
        calculate( Lp, [Lp1] ),
        substitute(L,Lp1,L1,Tree,Tp)
    ]
  ) :-
    equation( e( L1, R ), X ).
```

Fig. 6 Primitive operator.

expression satisfies the goal specifications (e.g., the left hand side is only a single variable and right hand side is a real number).

Figure 6 shows the form and an example of the primitive operator. A primitive operator consists of (1) target pattern (`Inquiry`) : the pattern of the expression against which the operator matches, (2) constraints (`PreConditions`) : the list of constraints which the target pattern has, (3) resulting pattern (`NextInquiry`) : the pattern which is translated by applying the operator, (4) operations (`PostConditions`) : the operations to translate the target pattern into resulting pattern.

The forward inference must make a search space explode. This system has the following constraints to prevent the explosion;

(1) constraint on the search time,

(2) prohibition of the reiteration of problem states,

†Henceforth, we use the term "operator" in place of "primitive operator"

(3) prohibition of the reiteration of differences in the problem state.

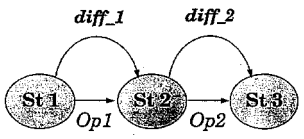
The first constraint limits the number of operators the inference engine can apply. If applying the operator results in a same state which is in the past of problem solving, the second one makes the application of an operator fail. By defining the difference between the state before and the state after applying an operator, the third constraint prevents the application of operators which make the same state difference as the last one (see Fig. 7).

5.3 Specialization of SCRs

The system specializes given SCRs (target concepts) based on the experiences of the problem solving (training examples). In general, the EBL generalizes training examples and produces new concepts. The variable is to what degree does the EBL generalize concepts. The most generalized concept is a target concept, and the most specialized concept is a training example, and the knowledge used to control the degree of generalization is called "bias".

The bias in this system is called a "proof schema". The proof schema specifies how the target concepts (i.e., SCRs) are specialized according to the examples of the solution. Figure 8 shows two examples of proof schemata, which suggesting the successions of the operator application (op_succeeds).

Each proof schema is in the following form; ps(*ID*, *TC*, *Body*). *TC* is the target concept and *Body* is the schema which represents how to specialize the target concepts. The system specializes *TC* by explaining the solution process with proof schemata. The term "explanation" here means the logical proof of *Body* by the proof schemata.



If *diff_1* is equal to *diff_2*
Then *Op2* is failed.

Fig. 7 The failure based on the state difference.

Figure 8 shows two proof schemata which specialize a target concept op_succeeds. One is used to represent a node which directly derives a solution, and the other represents a node which derives a node known to succeed.

The predicate `applicable( Node, Inq, Op )` means that at a node `Node`, a problem `Inq` satisfies the preconditions of an operator `Op`. The predicates `applicable` are automatically generated from the given primitive operators. The predicate `solves( Inq, Op )` means that a node, derived by applying an operator `Op` to a state `Inq`, satisfies the conditions to be a solution.

The predicate `child_node_after_applying_op ( ChildNode, ChildInq, Node, Inq, Op )` means that a node `ChildInq` is derived from a node `Node` by applying an operator `Op` to a state `Inq`. An operator `Op` translates a state `Inq` into `ChildInq`. This predicate causes a goal regression which is one of the most basic and important operations in EBL.

The specialization of SCRs are fulfilled as follows. Given an instance of the process of problem solving, which is a chain of states, the system chooses for each state a proof schema that has a body matched against the state. The system specializes the target concept (SCRs) by recursively specializing the formulas in a proof schema which explains the concept. The specialization of the formulae is accomplished by replacing the variables with corresponding values in target examples and renaming them, or by replacing the formula with another proof schema. The latter operation recursively calls specialization process, and this recursion terminates upon encountering primitive formulae. A primitive formula is the formula which has no proof schema.

The specialization technique (EBS algorithm) used by our system is well defined in [4]. Since the major

An operator (Op) succeeds if it directly solves the inquiry (Inq).

```
ps( os01,
    op_succeeds( Node, Inq, Op ),
    [
        applicable( Node, Inq, Op ),
        solves( Inq, Op )
    ]).
```

An operator succeeds if it succeeds after precursor operator is applied.

```
ps( os04,
    op_succeeds( Node, Inq, Op ),
    [
        applicable( Node, Inq, Op ),
        op_succeeds( ChildNode, ChildInq, NextOp ),
        child_node_after_applying_op( ChildNode, ChildInq, Node, Inq, Op )
    ]).
```

Fig. 8 The examples of the proof schema.

purpose of this paper is not to study an EBS algorithm itself, the details of the algorithm are not described here.

5.4 An Example of the Learning of SCRs

Given a problem, the following is a part of its solution path.

```
Node; Problem                              [Op]
-----------------------------------------
n3; equation(e([m(1,x)],[2,1]),A)   [op4]
n2; equation(e([m(1,x)],[3]),A)     [op7]
n1; equation(e([m(1,x)],[3]),3)     [finish]
```

The system learns the following SCR first (details of the specialization process is omitted here).

```
op_succeeds(N,equation(e([m(1,D)],[B]),C),op7)
[
  match_preconds([realp(B)]),
  known(N,[B=C]),
  solves(equation(e([m(1,D)],[B]),C),op7)
]
```

In this situation, the system specializes the schema os4 shown in Fig. 8, with the experience of applying the operator op4 to the node n3. After substituting the situations at node n3 for os4, the following clauses are generated;

```
op_succeeds(n3,equation(e([m(1,x)],[2,1]),A),op4)
[
  applicable(n3,equation(e([m(1,x)],[2,1]),A),op4),
  op_succeeds(CN,CI,NextOp),
  child_node_after_applying_op
  (CN,CI,n3,equation(e([m(1,x)],[2,1]),A),op4)
]
```

Substitute a clause op_succeeds for the one which has already been specialized at the child node of n3, and get the following clauses.

```
op_succeeds(n3,equation(e([m(1,x)],[2,1]),A), op4)
[
  applicable(n3,equation(e([m(1,x)],[2,1]),A),op4),
  op_succeeds(n2,equation(e([m(1,D)],[B]),C),op7)
  child_node_after_applying_op(
        n2,equation(e([m(1,D)],[B]),C),
        n3,equation(e([m(1,x)],[2,1]),A),
        op4)
]
```

Finally, the substitution of the predicate op_succeeds at node n2 and applicable at current node derives the following useful SCR.

```
op_succeeds(N,equation(e([m(1,B)],C),A),op4)
[
  known(N,[ partially_calculable(C,E),
            calculate(E,[H]),
            substitute(C,H,J)
          ]),
  match_preconds([realp(J)]),
  known(NC,[J=D]),
  solves(equation(e([m(1,B)],[J]),D),op7)
]
```

The acquired SCR here explains that the application of an operator op4 to the expression "$x = exp\_C$" succeeds, because the expression exp_C is partially calculable, a result (J) of calculation † of an exp_C is a real number, the replacement of a right hand side with the result of former calculation (J) causes an expression exp_J, and the application of an operator op7 to an exp_J satisfies the condition of a solution.

This SCR also shows a relationship between two types of knowledge where each of them belongs to EK and PK respectively. In terms of SCRs, the target concept implies the operational knowledge which belongs to the EK, and knowledge appearing in the body of SCRs belongs to the PK. In an above example, a knowledge of an application of the operator op4 to an expression "$x = exp\_C$" is the EK, and other knowledge in the body part (e.g., partially_calculable, calculate, etc.) is the PK.

Taking the SCRs as the knowledge of relationship between the EK and the PK, given the surface level student model which represents the understanding state on the EK, the system diagnose the students' understanding about PK. For example, if a student fails to apply the operator op4 to an expression "$x = exp\_C$", then this student might not understand that exp_C is partially calculable, and/or how to calculate it, etc. Furthermore, after diagnosing that particular knowledge in EK is not well understood by a student, the system can provide an instruction based on PK related to that knowledge.

6. Conclusion

This paper presented a student model diagnosing system which has the knowledge acquiring module to learn the knowledge of the relationship between *effective knowledge* (EK) and *principle knowledge* (PK). We discussed the learning module which acquires the search control knowledge via its own problem solving experience. The acquired search control knowledge represents the relationship between EK and PK.

Compared with the related works on student modelling, this approach has the following distinctive features. The knowledge of relationship between the EK and the PK is important to diagnose a student model in terms of PK (which is called deeper level student model [3]). One of the main contribution of this research is to reduce the effort of describing such knowledge by hand. By using the student model on the EK, and the acquired SCRs as the knowledge of relationship between EK and PK, the system can provide logical explanations according to the students' problem solving performances. These explanations are based on the knowledge, in the EK, which is not understood by a

† Indeed, the calculation of an exp_C is not a partial, because an expression might be the form of $A + B$ where both A and B are real numbers.

student, and are provided in terms of the PK. So, as a result and the second contribution of this research, the explanations are reasonable and easily understandable.

The search control rules handled here only suggest that a particular sequence of operators is preferred because it would make problem solving efficient. So, the system can only say that "you should memorize this operator (which belongs to EK), because you can solve problems faster with it" or something like it. In general, the human appropriately memorizes macro operators (EK) and solves problems in an effective way. That is, the human knows which operators are worth memorizing and which aren't. This decision seems to be justified by not only the time cost (i.e., efficiency) of the problem solving, but also by some other empirical reflections. It is a further problem of this research to identify a discriminative function which detects the educational worth of memorizing macro operators.

References

[1] Burton, R.R., "Diagnosing bugs in a simple procedural skill," In D. Sleeman and John Seely Brown, editors, *Intelligent Tutoring Systems*, pp.157–184, Academic Press, London, 1982.

[2] Ikeda, M., Mizoguchi, R. and Kakusyo, O., "Student mode description language smdl and student model inference system smis," *Trans. IEICE*, vol.J72-D-II, no.1, pp.112–120, 1989.

[3] Matsuda, N. and Okamoto, T., "Student model and its recognition by hypothesis-based reasoning in ITS," *Trans. IEICE*, vol.J75-A, no.2, pp.266–274, 1992.

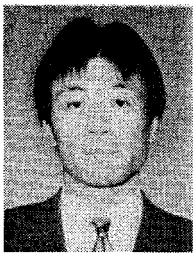
[4] Minton, S., *Learning Search Control Knowledge: An Explanation Based Approach*, Kluwer Academic Publishers, Boston, 1988.

[5] Okamoto, T., "The study on ITS for teaching the procedures of simplification in set-operations," *Trans. IEICE*, E73, no.3, pp.323–331, 1990.

[6] Self, J.A., "Bypassing the intractable problem of student modelling," In *Proceedings of ITS-88*, pp.18–24, 1988.

[7] Takeuchi, A. and Otsuki, S., "Formation of learner model by perturbation method and teaching knowledge," *Trans. of IPS of Japan*, vol.28, no.1, 1987.

[8] Wenger, E., *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann, Los Altos, CA, 1987.



Noboru Matsuda received B.S. and M.S. degrees from Tokyo Gakugei University in 1985 and 1988, respectively. He was a Research Associate in the Faculty of Engineering, Kanazawa Institute of Technology, from 1988 to 1993. He is currently a Research Associate in the Information Systems Laboratory, University of Electro-Communications, Chofu-shi, Tokyo, JAPAN. He has been interested in research on Intelligent Tutoring Systems, Educational Technology, and Knowledge Engineering. Mr. Matsuda is a member of JSAI, IPSJ, and CAI Society.

Toshio Okamoto graduated in 1971 from Kyoto University of Education and completed coursework in graduate school in 1975 at Tokyo Gakugei University. He obtained a Dr. of Eng. degree from Tokyo Instetute of Technology. Presently, he is a Professor in the Information Systems Laboratory, The University of Electro-Communications. He is engaged in research on AI model for intelligent CAI system. He translated *Artificial Intelligence and Intelligent CAI System.* He is a member of the Board, CAI Soc., and Jap. Soc. Educ. Tech. He also is a member of Committees, Educ. Eng., and Artif. Intel. & Knowledge Proc., IEICEJ, and a member of Soc. Artif. Intel.; AAAI; and Jap. Soc. Educ. Psych.