Learning through Intermediate Problems in Creating Cognitive Models

Kazuhisa Miwa

Graduate School of Information Science, Nagoya University, Nagoya, Japan

Junya Morita

School of Knowledge Science, Japan Advanced Institute of Science and Technology, Ishikawa, Japan

Ryuichi Nakaike

Graduate School of Education, Kyoto University, Kyoto, Japan

Hitoshi Terai

Graduate School of Information Science, Nagoya University, Nagoya, Japan

Correspondence author:

Kazuhisa Miwa
miwa@is.nagoya-u.ac.jp
+81-52-789-4747

Learning through Intermediate Problems in Creating Cognitive Models

Cognitive modelling is one representative research method in cognitive science. It is believed that creating cognitive models promotes learners' meta-cognitive activities such as self-monitoring and reflecting on their own cognitive processing. Preceding studies have confirmed that such meta-cognitive activities actually promote learning effects. However, there are some difficulties in bringing about learning by creating cognitive models in an educational context. To overcome the difficulties, we propose an innovative learning design, "learning through intermediate problems", and also developed a web-based production system called DoCoPro that can be used anywhere and anytime in an environment connected to the Internet. We performed three introductory cognitive science classes in which the participants learned cognitive modelling and constructed running computer models using our system. In the first and second class, the participants were required to construct production system models that solve pulley problems. They also posed their original pulley problems that their own models were subsequently able to solve. These generated problems were distributed to the other members. The participants were able to find incompleteness in their cognitive models, revise them to remove the incompleteness, and improve their models while solving the given problems. The participants, by successfully creating sophisticated models, acquired a deeper knowledge of the learning domain. The class practices confirmed the utility of "learning through intermediate problems" when constructing an educational environment for learning creating cognitive models. In the third class, the participants constructed cognitive models solving addition and subtraction problems using DoCoPro. The cognitive processing underlying such problem solving is automated, therefore it may be difficult to verbalize and externalize such cognitive processes. The post questionnaire showed evidence that the participants actually performed meta-cognitive activities while monitoring their own internal information processing.

Keywords: word; Cognitive modelling, Production system, Reflection, Web-based application

Introduction

Model-based approach in cognitive science

The model-based approach is one main research method in cognitive science. This innovative research approach opened a new era in the history of studies for exploring human intelligence. Schunn et al. (1998) examined the manuscripts published in Cognitive Science, an official journal of the cognitive science society, and concluded that the model-based approach has taken a central role in the studies of cognitive science. There are many frameworks for constructing cognitive models; rule-based description has been widely approved as one of the most typical frameworks. Production system, initially developed by Newell and Simon (1972), is a popular framework for embodying the rule-based description in computer programs. In recent cognitive science studies, some standard architectures, such as SOAR (Newell 1990) and ACT-R (Anderson & Lebiere 1998), have been widely used in the cognitive science community.

Advantages of cognitive modelling

There are many advantages in building computational models of cognition. The following is a list of the advantages that almost all cognitive scientists accept (Fum et al. 2007).

- Clarity and completeness
- Better exploration and evaluation
- Serendipity and emergence

In terms of the clarity and completeness, verbally expressed statements are sometimes flawed by internal inconsistencies and are also incomplete because

unconscious processes cannot be reported. To build a properly representative computer program, researchers have to describe accurately every process occurring in the mind. Computational modelling forces the hidden assumptions not reported to become fully explicit (Cooper et al. 1996).

From the viewpoint of learning activities, this characteristic of constructing cognitive models may bring important educational advantages. To construct cognitive models, learners have to understand what processes underlie the solving of a specific task. They may be led to focus on their own cognitive processing to find such processes. This activity can be expected to activate learners' reflective thinking or meta-monitoring of their cognitive processes.

Meta-cognition includes a wide variety of cognitive activities. Two dimensions are decisively important to categorize such meta-cognitive activities. One dimension is monitoring either memory or processing, and the other is either ongoing or retrospective activities. When constructing cognitive models in our contexts, participants have to understand what procedures underlie the solution processes of a specific task. They are expected to be led to focus on their own cognitive processing to understand such processes. Therefore, the meta-cognition that we intended to focus on in this manuscript is monitoring on-going cognitive processing. When creating cognitive models, learners are guided to monitor their thinking processes for formalizing their knowledge; i.e., they formalize knowledge based on the rule description framework for constructing cognitive models to run on computers. Such formalization may elaborate their knowledge and promote deeper understanding of learning domains. Many preceding studies have confirmed that such reflective thinking and meta-cognitive activities promote learning (Renkl 1997; Renkl et al. 1998). One representative way to promote monitoring activities is self-explanation (Chi et al. 1989; 1994). Based on their

findings, many educational systems, such as Lisp Tutor (Pirolli and Recker 1994), Geometry Cognitive Tutor (Aleven and Koedinger 2002), and SE-Coach (Conati and VanLehn 2000), were developed for improving learning by activating such self-explanation activities.

Difficulties in cognitive modelling

Even though there are many advantages in cognitive modelling, there are problems to be solved in computer modeling when having novice learners try to construct successful running models. One big issue is the usability of models (Pew and Mavor 2007; Ritter 2009). Usually, expert skills are needed to construct computational models. Additionally, computational models when expressed as computer programs are not easily understood by other persons than the person who writes the program. It is relatively difficult for group members, especially novice learners, to share computer models they have created. In an educational context, this problem is crucial.

Almost all production system architectures available have been developed as research tools for expert users. There are some trials to improve the usability of models. One approach is to develop higher-level languages that may be easily used by novice users. Usually compilers that translate such higher-level languages into the representations of the established architecture models such as Soar and ACT-R are also developed. One of the earliest trials is TAQL (Yost 1993); and many languages have been established. Ritter and his colleagues reviewed recent trials, and indicated that those languages actually successfully shorten the time of model construction (Ritter et al. 2006). COGENT is another trial (Cooper 2002). COGENT is a computational modeling environment within which information processing models of cognitive processes have been developed with a graphical user interface. It has several features that simplify the processes of creating cognitive models and ease the difficulty of

reading computer models. They are still technical systems for experts and relatively difficult for novices to use.

There are some preliminary trials on the education of novice learners in computational cognitive modeling (e.g., Sewart 2004). One educational system developed in this context for novices is Hank, a visual programming environment for cognitive modeling (Collins and Fung 2002). Hank was developed for undergraduate students; and it basically consists of the spreadsheet-based database and the retrieval mechanisms using cue information. Hank simulates just memory retrieval processes rather than problem solving processes, and is used for having learners understand basic cognitive theories such as the schema theory. Therefore, it is not useful to let users reflect their cognitive processing.

Many cognitive science classes worldwide are dealing with the issues of cognitive modeling. However from the above difficulties, it seems that not so many classes exist where participants actually construct running computer models and experience such excitement and effectiveness.

There is another practical difficulty from the viewpoint of teachers. One big practical obstacle is how to construct an educational environment. Usually, the behavior of each type of cognitive architecture depends on a lower computer platform than that on which the architecture has been installed. It is no pleasure to install cognitive architecture into many computers for class practice.

To overcome such difficulties, we developed a web-based production system architecture called DoCoPro, which is a nickname taken from a Japanese phrase that means a production system for anywhere. Our architecture has been developed especially for novice users. Therefore our architecture provides novice learners with a graphical user interface directly manipulated and multiple learning support that reduces

the difficulties that emerge in the process of programming. Learners can use DoCoPro whenever and from anywhere with access to the Internet. Our system can be used on a web browser without any specific software. Instructors are completely released from the frustration of preparing an educational environment.

Interaction design for learning cognitive modeling

Another big issue for learning by creating cognitive models is instructional design. Recently, in learning sciences, collaborative learning has taken a central role as the standard design for learning. In learning activities, learners are not given knowledge by teachers, but construct knowledge by themselves through collaborative activities among learners. Therefore interaction design is a crucial issue in computer supported collaborative learning (CSCL). Interaction design is defined as design of learning environments such as learning materials, learning systems, instructors, and group construction of participants, for activating interaction of participants for constructive learning. Multiple guidelines and methods for the interaction design have been proposed (Gros 2001; Strijbos 2004; Isotani 2009).

One of the key factors for successful collaborative learning is the idea that the differentiation of knowledge among learners gives them the cues to accept other members' knowledge, and promotes their learning activities. For example, in jigsaw learning (Aronson and Patnoe 1996), first, expert groups are constructed within which all participants focus on and learn a piece of specific knowledge becoming experts on it. Then the jigsaw groups are reconstructed such that the participants in the expert groups are rearranged. Various experts are involved in each new jigsaw group and the participants promote learning while unifying various kinds of knowledge through interacting with other experts of other domains. The jigsaw method has been widely accepted in the learning science community (Miyake and Shirouzu 2006). Additionally,

it has been confirmed that not only the differentiation of knowledge of community members but also the heterogeneity of learning materials function well in CSCL (Fidas et al. 2005). In jigsaw learning, participants notice the incompleteness of their knowledge through interaction with other expert members, and try to accept other members' perspectives and knowledge, causing active interaction. In our approach, we intend to bring about this type of activation for interaction in learning cognitive modeling by overcoming some difficulties mentioned later.

Heterogeneity of knowledge and materials available can be expected to motivate and necessitate interaction among members. However, heterogeneity does not necessarily bring about the necessity and sufficiency of effective learning. Learners may face difficulties in referring to other members' knowledge. They do not know who has the most helpful knowledge. Knowledge awareness is a central issue in successful collaborative learning (Dourish and Bellotti 1992; Ogata and Yano 2000).

In learning cognitive models, this kind of activity corresponds to the activities in which participants learn the advantages of others' models and, based on the understanding of differences between their own model and others' models, revise their own model to remove its disadvantages. However, it is quite difficult for novices to compare their model with others' models directly. Novice learners suffer from reading program codes that have been written based on the framework of the production system, and feel difficulty to understand knowledge as represented in codes, which we have already referred to as a problem with the utility of models.

Learning through intermediate problems

We address these difficulties by proposing an innovative instructional design called "learning through intermediate problems".

Nature of Learning through intermediate problems

In the class activities, a learning context is created where interaction among class members emerges naturally in the learning processes, and participants are guided to learn cognitive modeling in a social context. Figure 1 shows the concept of LtIP (learning through intermediate problems). Each learner constructs a cognitive model, and also poses a problem that his/her model can solve. Each problem is opened to the other members. This means that each learner acts not only as a problem solver but also as a problem poser.

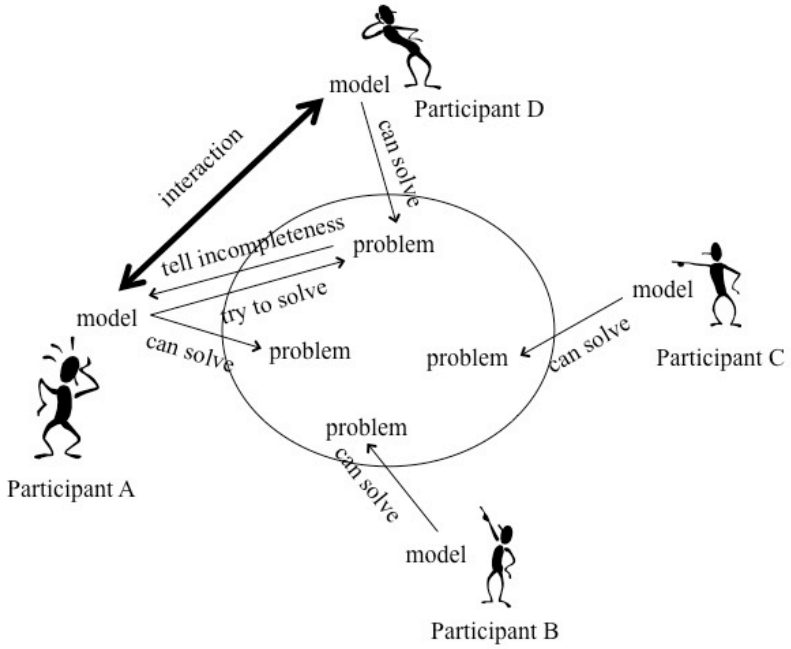


Figure 1: Concept of learning through intermediate problems.

Each participant has his/her model solve the problems posed by the other members. The differences between his/her own model and the other models are recognized while solving various problems from others. Knowledge for solving each problem is incorporated in each model; therefore participants are naturally guided to become aware of the differentiation of knowledge incorporated in the models created by

their group members. For instance, when one model cannot solve a certain problem, the participant knows that the knowledge incorporated in another model that is able to solve the problem is absent from his/her model. The incompleteness of his/her model emerges by solving the problems given by the other members. It has been confirmed that the feedback of negative information telling incompleteness and inadequacy improves the performance of learning and discovery. LtIP is an instructional framework for generating the interaction through which this kind of negative information naturally emerges. Improving a model means making up for the differences among the models. LtIP generates interaction among the cognitive models through a problem as a mediator, overcoming the difficulties in making direct comparisons among models.

There are several specific factors in LtIP. First, interaction emerges through intermediate objects, i.e., problems in this case. Participants do not interact with each other directly. Interaction occurs implicitly not explicitly (Hansen et al., 1999). We tested the utility of this learning design in our university classes reported in the following. In the situation, the participants simultaneously learn in an identical classroom. However, the participants' interaction occurring in the classroom is not synchronous but asynchronous, meaning that each participant tries to solve problems that were posed by other participants in advance. The similar situation could be established by a set of problems that were provided by an instructor. However, in LtIP, a natural context of competition and interaction emerges by mutually solving the problems posted by the group members.

Related studies

There are some trials in which problem-posing activities were brought about in educational contexts. Hirashima and his colleagues developed a learning environment,

called MONSAKUN, where learners generate arithmetic word problems by selecting and ordering simple short-unit sentences (Hirashima et al. 2008; Hirashima et al. 2011). The participants voluntarily engaged in problem posing in a real class setting. The results showed that their system improved the problem solving abilities of lower performance students. It has been demonstrated that problem posing is effective especially in mathematics education (Ellerton 1986; Silver & Cai 1996; English 1997); however, it has also been indicated that participants tend to generate only standard prototype problems (Kojima & Miwa 2010). Kojima and Miwa developed a learning environment that enables learners to generate a variety of problems by systematically presenting problem examples (Kojima & Miwa 2008). These approaches basically focused on problem posing. On the other hand, in LtIP, problem-posing is a secondary activity. LtIP intends that problems function as a mediator for interaction through which participants' cognitive models are improved.

In computer science education, problem construction activities were also introduced. Gotel developed a pedagogical approach where students formulate problems, code programs for solving the problems, and contribute their programs with test cases to an open source web-based system, WebWork (Gotel 2008). Even when the problem topics taken by the participants were relatively restricted, it was expected that the participants would more carefully test their programs with test cases because they expected that their programs would be re-used by other members in the group, maybe imagining various situations in which the programs would be used. In their approach, the main focus was on problem formulation and program coding activities. There was no feedback phase in which initial programs were reconstructed through interaction with others, whereas in the LtIP program, reconstruction caused by interaction with other programs through intermediate problems was mainly focused on.

The above approach basically focused on the improvement of participants' programming skills. On the other hand, LtIP intends to have participants learn reflective thinking and understand deeper domain knowledge. Biswas and his colleagues developed a learning environment in which participants created a concept-map-based model of a learning domain. The participants did so by teaching a teachable agent, called BETTY (Biswas et al. 2005; Biswas et al. 2010). Schwarts and his colleagues defined two types of meta-cognition: self-directed meta-cognition and other-directed meta-cognition (Schwarts 2009). They pointed out that it is usually difficult for naïve students to engage in meta-cognitive activities because participants must engage in dual tasks: problem solving activities on the cognitive level and monitoring and controlling activities on the meta-cognitive level. The other-directed meta-cognitive activity is useful because each of the dual tasks may be assigned to either a participant or a teachable agent; participants can concentrate only on meta-cognitive activities. Learning by teaching is effective in acquiring domain knowledge with active meta-cognitive activities. However, this approach is performed based on the "learning by teaching" framework. On the other hand, in creating cognitive models in LtIP, participants constitute their cognitive model, directly monitoring their own mental processing and problem solving processes.

Beginners are usually required to construct a basic cognitive model that solves a simple example problem in the initial stage of learning to understand the grammatical structures and the semantics of the programming language. It usually takes a long time to acquire technical skills for cognitive modeling; therefore, learners are forced to engage in difficult practice requiring much patience in the introductory phase of learning. A similar problem was pointed out in creating intelligent tutoring systems. Koedinger and his colleagues proposed "Pseudo Tutors" in which programmers encode

abstract procedural programs rather than production rules to reduce such time-consuming training for coding AI programming (Koedinger 2004). They successfully shorten the time for creating tutoring systems. However, the abstract level of knowledge is crucially limited in such programming so that developers have to explicitly code an individual procedure to respond to every specific situation. This limitation is crucial in our learning contexts; therefore, we try to have participants encode production rules, and LtIP was introduced to overcome the difficulties.

Cognitive modeling has been widely approved of as a strong research tool for exploring the human mind. However, there are not many empirical studies for understanding what modelers as learners acquire by creating cognitive models. Experience in constructing cognitive models is expected to facilitate various aspects of learning: e.g., the acquisition of programming skills, learning of psychological mental processes, and an understanding of the domain knowledge. It is important to distinguish learning *by* creating cognitive models from learning *of* creating models. When we use cognitive modeling as an educational tool, the former learning is a central issue. The latter is important for cognitive science major students, but not for non-major and naïve students. Learning of programming skills is involved in the latter while understanding of the nature of human information processing and deepening of understanding in a learning domain are involved in the former. In this paper, we investigate multiple aspects of effectiveness of "learning by creating cognitive models" using log analysis, a questionnaire, and a posttest.

DoCoPro: Production System for Anywhere

Our system is called DoCoPro, which is a web-based production system based on the server and client model. Since DoCoPro can be flexibly utilized in various learning

environments, teachers can easily introduce the system into classes. Learners can use it just by accessing the server through such standard web browsers as IE (Windows), Safari (Macintosh), and Firefox (various OSs). DoCoPro can be used without any preparation such as installing the architecture itself and related sets of software because the architecture runs on a web browser. Practically, this has a big impact in real educational settings. In addition, since the server handles most of the information processing and the web browser on the client's computer only handles the operation of the user interface, such as user operations and screen displays, even low performance old personal computers can also be used. Therefore, our system can be used in various types of classroom, such as computer rooms in which both old and new personal computers co-exist and standard classrooms into which private notebook computers, whose operating systems and information processing performances are largely different, are brought.

Managing data in a unified manner using its database is one of the most important features of DoCoPro. Since each user's individual circumstance is held in the server by the login mechanism with password authentication, he/she can engage in continuous learning in individualized study at home after learning in school under identical learning circumstances.

On the production system architecture, procedural knowledge is implemented as production rules and declarative knowledge is stored in working memory. To construct a cognitive model, modelers formalize their knowledge as if-then rules, and confirm that successive applications of the rules transfer the working memory from the initial to the goal state. Therefore, the cycle of rule construction, rule execution, and rule editing is a central activity in cognitive modeling. The synopsis nature of the viewed screen has been stressed in the design of the system interface. All information is displayed in a

single window (see Figure 2). The production rules displayed on the right side of the window can be directly manipulated (e.g., editing, inserting, deleting, and sorting) without a text editor. The left-center controller button forwards inferences promptly during editing. System messages, such as learning support information explained in the following, are displayed in the left-lower window. The contents of the working memory are updated by developing the inferences. DoCoPro does not distinguish the editing mode where rules are edited and the execution mode where rules are performed. This feature simplifies debugging rules and decreases the cognitive load of learners.

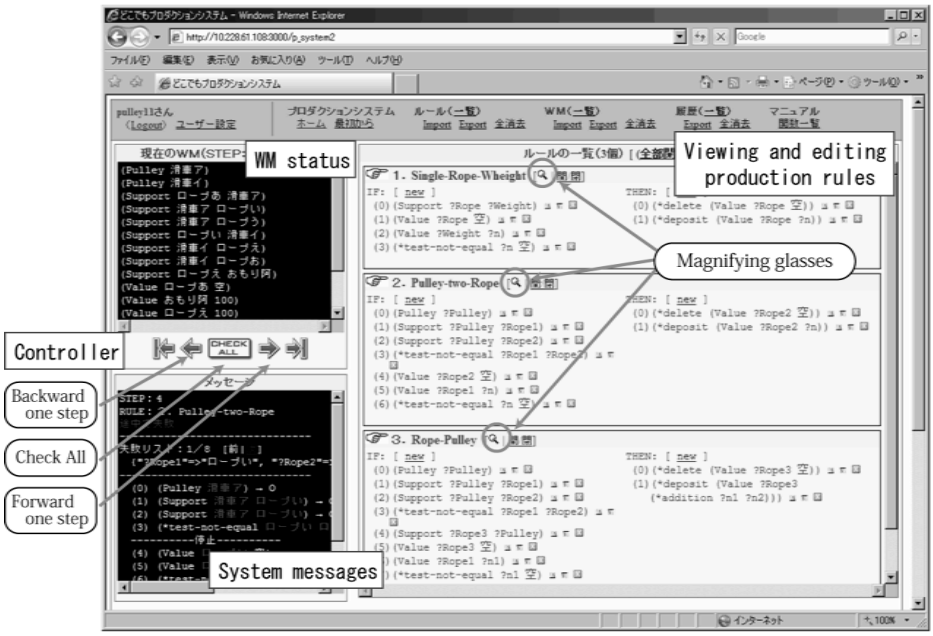


Figure 2: Screen shot of DoCoPro interface.

The online modification function enables users to make steady progress when developing inferences in a step-by-step manner. Online modification provides users with flexible rule editing; i.e., users can edit each rule while temporally stopping the inference process, and restart the inference from the step after editing is completed.

DoCoPro can also retrace the inference process to any preceding step. This process management and the functions for flexible rule editing mentioned above enable learners to edit rules whenever they discover errors, and restart the inference from any step.

One learning support feature of our system is the presentation of hints that assists understanding of the recognize-act cycle of the production system, which includes rule matching with variable bindings, conflict resolution, and rule execution. The hint presentations have two functions: one indicates the candidate rules that can fire (activated by the "Check All" button in Figure 2), and the other provides detailed information about the matching status of the rules and the working memory (activated by the "magnifying glass" button displayed beside each rule in Figure 2). See Nakaike, et al., 2009a and Nakaike, et. al., 2010b for the detailed explanation and empirical evaluation of the utility of this hint presentation mechanism.

Class Practice 1

We designed and performed introductory cognitive science classes based on LtIP with DoCoPro, and discuss the utilities of the instructional design for learning cognitive modeling based on the class practice. A part of Practice 1 was already reported in Miwa et al. (2009) and Morita et al. (2009).

Task and participants

The task used in our class activities was pulley systems. The reason for using pulley systems is that many types of problem can be considered in which the perceptual features of problems and problem structures can be independently manipulated. The subjects and the cognitive models to be constructed are based on Larkin and Simon (1987). Figure 3 shows an example pulley problem and the knowledge required for

solving the problem.

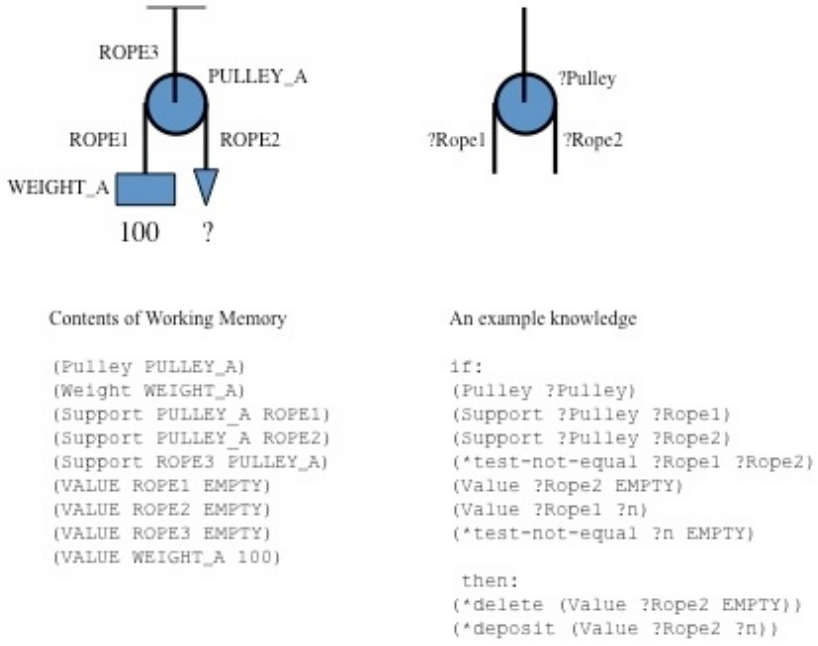


Figure 3: An example problem with its working memory representation and an example piece of knowledge for solving the problem.

Two classes participated in our practice.

- Class A: Eighteen liberal arts undergraduates without advanced programming skills.
- Class B: Fourteen information science graduate students. Some had experience with computer programming; however none had learned list-processing-type computer programming such as LISP and Prolog.

Class activities

Our practice was performed in a series of cognitive science introductory courses. The following is an overview of our class activities. Interaction among the participants emerged based on the LtIP schema presented in Figure 1. Almost all participants did

not only engage in the task in the classroom but also outside of the class such as homework using DoCoPro accessed by the Internet.

- Introduction: A lecture on the basics of production system modeling was given to the participants. They received a simple pulley problem and were instructed to construct a production system model that solves the problem.
- Constructing Initial Models: All participants were required to generate an original pulley problem and to construct a cognitive model to solve a problem they had generated by themselves. The models constructed in this earlier stage are called initial models.
- Constructing Problem Set: The generated problems were collected, and a set of problems was created. The problem set consisted of 14 problems in Class A and 15 problems in Class B.
- Initial Model Challenges: The problem set was distributed to the participants, and their initial models were used to try to solve the problems. The solved and unsolved problems were identified for each of the initial models.
- Constructing Revised Models and Challenges: The participants were required to improve their initial models by having their models solve as many problems as possible. The improved models are called revised models. The revised models were again used to try to solve the problems of the problem set, and the solved and unsolved problems were identified.
- Final Task: In the final stage of the class, the participants constructed original models. The participants could select a final task by themselves. Some might select it from a cognitive science textbook, and others might focus on a familiar topic from daily life. For example, the Rubik's Cube, the Missionaries and Cannibals Problem, and the Three Prisoners Problem were chosen as problems

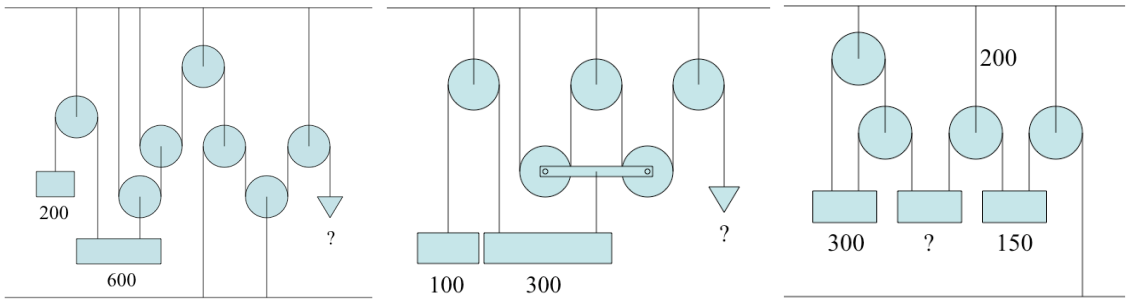
for the final task. The important point is that they chose a relatively complex task as a final task. The means of the numbers of rules are 10.3 in the initial models, 24.8 in the revised models, and 45.1 in the final models, supporting that the complexity of models increased from the initial to final models. The most complex model contained 170 rules and completed the six sides of the Rubik's cube in about 1000 steps.

In the introduction of the problem generation phase, a teacher gave the participants the following instruction: generate a possible difficult problem that your model can solve but other members cannot solve. This context brought about a competitive situation in the class activities. As they tried to generate a more difficult problem to beat the others, they had to create a higher performance model that could solve such a difficult problem of their own. Basically, the participants were not instructed to create sophisticated models; however, such a competitive setting guided them to do so. This setting naturally produces interaction among the participants and increases their motivations to construct sophisticated models.

Overall Results

Collected Problems

In Class A, some of pairs of the participants collaboratively proposed one single problem. In Class B, one of the participants proposed an invalid problem. As a result, 14 problems in Class A and 15 problems in Class B were collected. The problems collected were grouped into three categories: simple problems, specific problems, and difficult problems. Figure 4 shows example problems.



(a) Simple problem (b) Specific problem (c) Difficult problem

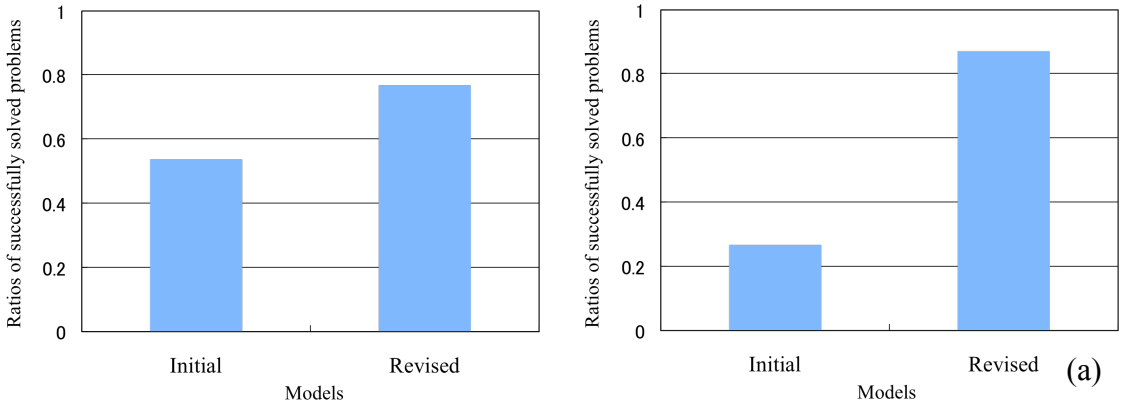
Figures 4: Example problems in problem set.

- Simple Problems: These problems were solved using a set of basic knowledge, such as the two forces acting on two ropes hanging with a pulley are identical; the force pulling up the pulley is identical to the sum of the forces of two ropes dragging the pulley down.
- Specific Problems: These problems contain specific objects in the system. In the example problem in Figure 4(b), there is a rod connecting two of the pulleys. These problems are solved using specific knowledge applicable to a specific situation.
- Difficult Problems: These problems are difficult to solve because each of the two forces acting on the two ropes supporting each weight is not calculated independently from the other; therefore the ratio of the two forces has to be inferred for problem solving. In the specific problems, a single rule corresponding to a specific situation makes the solution possible; however in the difficult problems, multiple relatively complex rules are needed for gradual inferences to calculate the ratio.

In Class B, seven simple, four specific, and four difficult problems were collected. In Class A, twelve simple and only two difficult problems were collected; one invalid problem was eliminated. The result showed that all students relatively easily generated pulley problems. However, in Class A, most participants could generate only simple problems, whereas the participants in Class B successfully generated a variety of problems. Therefore, effective problem generation may depend on programming skills or domain knowledge.

Model Performance

Figure 5 shows the mean ratios of the problems successfully solved by the initial and revised models to all problems, indicating that the models' performances largely improved in both Classes A and B. An ANOVA showed significant differences of the models' performances between the pre and post phases both in Classes A and B [$F(1, 12)=21.27$, $p < .01$ in Class A, and $F(1, 12)=118.33$, $p < .01$ in Class B]. This supported the initial evidence that LtIP functioned well. The participants were guided to be aware of the differences between their own models and others', and improve their models.



Class A (b) Class B

Figure 5: Model Performance

Compared to Class A, the improvement in performance was larger in Class B. As mentioned earlier, various types of problem were generated in Class B. Therefore, the initial models could not solve such unseen problems; the ratio of successful problem solving was considerably low in the early stage. This explains the large improvement in Class B.

Types of models

In the following, we mainly analyze the results of Class B in detail because a large variety of problems were collected. Table 1 shows the combination of the revised models and the problems that each model could solve. "1" means a successful solution whereas "0" means unsolved. The models (vertical axis) are sorted based on their problem solving performance, i.e., the number of problems that the model could solve. The problems (horizontal axis) are categorized as simple, specific, and difficult, as mentioned earlier. The overall pattern of the arrangement of "0" is gathered in the right lower side. This means that the poor performance models could not solve the difficult problems.

Table 1. Problems each revised model could solve. The high performance models successfully solved all problems, the medium performance models solved some of the difficult problems, but did not solve some of the specific problems, and the low performance models solved none of the difficult problems.

		Simple							Specific				Difficult				
		Prob. 1	Prob. 2	Prob. 3	Prob. 4	Prob. 5	Prob. 6	Prob. 7	Prob. 8	Prob. 9	Prob. 10	Prob. 11	Prob. 12	Prob. 13	Prob. 14	Prob. 15	
High	Model A	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	15
	Model B	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	15
	Model C	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	15
	Model D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	15
	Model E	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	15
Medium	Model F	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	14
	Model G	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	14
	Model H	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	13
	Model I	1	1	1	1	1	1	1	0	0	1	0	1	1	0	1	11
	Model J	1	1	1	1	1	1	0	1	1	0	1	1	0	1	0	11
Low	Model K	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	11
	Model L	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	11
	Model M	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	11
	Model N	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	10
		14	14	14	14	14	14	13	13	13	13	11	10	9	8	7	

The revised models are categorized based on the solution patterns of the three types of problems.

- High performance models: solved all problems successfully.
- Medium performance models: solved some of the difficult problems, but did not solve some of the specific problems.
- Low performance models: solved none of the difficult problems, but almost all of the specific problems.

Table 2 shows the transition pattern from the initial to the revised models. One participant did not present his initial model; therefore a total of thirteen models were analyzed.

Table 2: Transition from initial to revised models in Class B.

		Revised			
		Low	Medium	High	
Initial	Low	4	5	1	10
	Medium	0	0	3	3
	High	0	0	0	0
		4	5	4	

The table shows that no participants created high performance models, and most participants (10 out of 13) created low performance models in the initial phase. Nine of thirteen participants improved their models: five from low to medium, one from low to high, and three from medium to high. This indicates that most participants actually improved their models during the class activities. In addition, the participants in Class A did not generate the specific problems containing specific objects. Therefore, their models were categorized into two groups: low and high performance models. All thirteen initial models were categorized in the low performance models; three of the

thirteen models were moved into the high performance models. Seven simple problems in Class B and twelve simple problems in Class A were generated. In Class B, thirteen out of fourteen models completely solved all simple problems. The other model could solve six of seven problems. On the other hand, in Class A, seven of thirteen models solved all simple problems, and three models solved ten of twelve simple problems. However, three models could solve only half of the basic problems. This difference between Class A and Class B seems to come from differences of programming skills of the participants.

Detailed Analysis

Log analysis

To confirm the development of rule based programming skills, we analyzed the log data stored in the DoCoPro server. In this analysis, requests to the server were divided into three manipulation types: Coding (coding of models), Step (running models), and Hint (viewing hints). We examined the transitions of the numbers of requests for these three manipulation types.

Figure 6 shows how many requests emerged in the period of the three tasks for the three manipulation types in Class B. The transitions seem different for each of the three manipulation types. To investigate the details of the difference, a 3 (manipulation types (within): Coding, Step and Hint) x 3 (tasks (within): the initial task, the revised task, and the final task) ANOVA was conducted with the number of requests as a dependent measure. The initial task means constructing initial models, the revised task means improving initial models and constructing revised models, and the final task means constructing original models that solved problems they themselves selected.

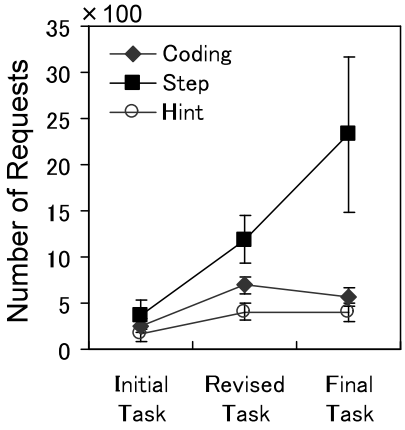


Figure 6: Results of log analysis.

As a result, both the significant main effects of manipulation types [$F (2, 24) = 5.28$, $p < .01$] and tasks [$F (2, 24) = 5.08$, $p < .01$] and the significant interaction of the manipulation types and tasks [$F (4, 48) = 3.20$, $p < .05$] were detected. While the simple main effects of the tasks were significant for Coding [$F (2, 24) = 5.53$, $p < .01$] and Step [$F (2, 24) = 3.93$, $p < .01$], the simple main effect of the tasks was not significant for Hint [$F (2, 24) = 1.63$, n.s.]. Furthermore, multiple comparisons (LSD) confirmed the increase of Step from the initial to the final tasks ($p < .05$), and the increase of Coding from the initial to the revised tasks ($p < .05$).

The above results reveal the characteristics of each task in the modeling processes. Since the frequency of viewing hints is proportionally large in the initial task, the process in this task can be considered to contain many trial-and-error behaviors in which the participants had frequently encountered impasses. On the other hand, in the revised and final tasks, the participants were able to construct models without the support of hints. Furthermore, the manipulations in the final task, in which Step occupies a proportionately large space, suggest that the participants finally became able

to construct complex models performing long problem-solving steps. These results confirm that the participants acquired programming skills by LtIP.

Posttest

Next to verify the development of understandings in the learning domain a posttest was performed in the final stage of the practice.

Material

The posttest consisted of ten pulley problems. The problems can be categorized into three types from the viewpoint of their problem structures.

- Simple Problems w/One Rope: The problems contain only weights supported by a single rope.
- Simple Problems w/Two Ropes: The problems contain weights supported by two ropes. However, the force acting on each rope is independently calculated; therefore the addition of a single relatively simple rule enables the models to solve such problems. An example is the problem indicated in Figure 4(a).
- Difficult Problems: Some of the problems collected in Classes A and B were defined as difficult problems. The difficult problems used in the posttest have the same features as those. An example of a difficult problem is indicated in Figure 4(c). To solve the problems, the ratio of the forces acting on the two ropes supporting a weight must be calculated.

The perceptual superficial characteristics of the problems, such as the number of pulleys involved in a problem, were manipulated independently from the problem structures characterizing the three types of problems. For example, some of the problems contained three pulleys and others eight pulleys. In two problems there was a

rod, connecting the pulleys. Though the perceptual features of these problems were relatively different from others, they can be solved only by a set of basic knowledge; therefore they are categorized into the simple problems.

Tasks

Two tasks were given to the participants.

- Problem solving task: The participants were required to calculate the force acting on a target rope.
- Categorization task: The participants were required to categorize ten problems into three categories. In this task, the participants were instructed to gather similar problems in each category. If the participants categorized the problems based on the problem structures, the three problem groups indicated in the material section were constructed.

Result

For the problem solving task, all participants except one accurately solved nine or more of ten problems, but for the categorization task, the results varied among the participants.

Analysis of the result of the categorization task was performed excluding one participant whose performance of the problem solving task was extremely low. The number of problems that didn't follow the normative categorization defined above was analyzed. Figure 7 shows the mean number of such non-normative categorizations into each of the three types by the participants who constructed high, medium, and low performance models. An ANOVA revealed that the main effect reached significance ($F(2, 9)=7.25$, $p<0.05$). A LSD analysis showed that the numbers in the low and medium models were larger than that in the high model ($p<0.05$ and $p<0.05$). The result implies

that the participants who constructed the high performance models followed the normative categorization, but other participants tended to perform non-normative categorization.

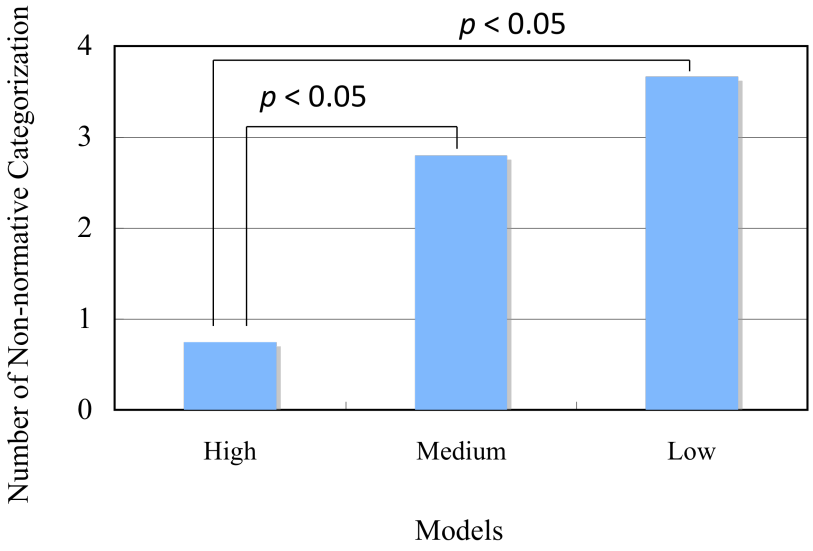


Figure 7: Result of Categorical Test

Discussion of Practice 1

Nature of learning through intermediate problems

The results of the first class confirmed that the participants successfully acquired skills of rule-based programming, and creating a sophisticated model improved understanding of the domain knowledge, insisting the utilities of LtIP proposed in this current study. We discuss the nature of the learning design.

It has been confirmed that the feedback of negative information improves the performance of learning and discovery from various perspectives such as unexpected findings (Dunbar 2001), surprising results (Kulkarni and Simon 1988), anomaly (Darden 1992), and falsifications (Miwa 2004). In learning cognitive modeling, learners are also expected to acquire various types of knowledge by facing the

limitations of their models and overcoming them. However, people tend to collect positive instances consistent with their beliefs and hypotheses (Klayman and Ha 1987). This implies that investigating the methods of providing learners with adequate negative information about the limitations of their models is important.

In the current practice, we designed class activities where interaction among the participants naturally emerges, and through the interaction each participant discovered his/her model's limitations by receiving negative information from others. Figure 5 shows that the performance of the revised models was drastically improved from the initial models. This evidence suggests that the instructional design proposed in this practice functioned well.

Understanding of learning domain

Note that the manner of problem categorization by each of the participants was largely different even though the performance of the problem solving task was almost identical for every participant. The result shows that the participants who constructed high performance models categorized problems based on the problem structures, and were not confused by their perceptual appearances. Chi et al. reported that in physics, experts categorized problems based on the structures of problems, and novices tended to do so based on appearances (Chi et al. 1981). The result of our practice indicates that the participants constructing high performance models successfully acquired high quality knowledge consistent with experts. Note that it remains unknown whether such knowledge acquisition comes from the experiences of creating sophisticated models, or they could create high performance models because they initially understood such knowledge. The investigation of such a causal relation is future work.

As mentioned earlier, the addition of specific rules corresponding to individual situations makes it possible for the specific problems to be solved by the basic models.

In general, the difficult problems needed more complex rules to be solved. However, in some cases, the addition of simple rules also gave the models the ability to solve the difficult problems. For example, see the example problem in Figure 4(c). When the following specific rule that can be applied only to this specific situation is added to a basic set of knowledge, the model is able to solve this problem.

If

The relationship among the two pulleys on the left side, the leftmost weight, and the ropes connecting those objects shown in Figure 4(c) is detected

Then

Let the ratio of the forces acting on the two ropes be two to one.

Actually some participants handled the difficult problems by describing such specific rules. Five models could solve the difficult problems in Class B. One of the five models consisted of specific local rules, each of which addressed one specific problem. On the other hand, in Class A, both models that could solve the difficult problems consisted of such local rules. In the current practice, no difference in performance in the posttest was detected between the participants who handled the difficult problems by adding specific knowledge and those who did so by producing more complex rules that could be generally applied to various situations. From the viewpoint of quality of learning, the relationship between learning performance and the manner of representing rules is crucial. Further investigation of this issue remains important future work.

Class Practice 2

The first class practice confirmed that LtIP functioned well for learning rule-based programming and understanding domain knowledge. As mentioned in the introduction another important function of creating cognitive models is to activate participants' reflective thinking. To construct cognitive models, modelers have to describe human cognitive information processing formally. To do so, they must monitor their own thinking processes. It is expected that such experiences of creating cognitive models lead learners to understand human cognitive information processing more deeply, and reflect more seriously on their own procedures for problem solving, which they were initially not conscious of. To verify this point, we performed an additional class practice.

Task and participants

Tasks dealt with in the class were addition and subtraction problems. Undergraduate students in Japan participating in this class routinely solve these problems. It is assumed that cognitive processing for solving these tasks may be automated, and it may be difficult for them to verbalize and externalize the cognitive processing occurring in the mind (Ericsson and Simon 1980). Therefore, the tasks are suitable to confirm the effects of introspective activities. Additionally, it has been confirmed that the mental procedures in the tasks were suitable for production system modeling, and naturally well formalized based on the rule based description (Brown and Burton 1978; Young and O'Shea 1983).

Eighteen participants joined the second class practice. To confirm the effect that creating cognitive modeling actually activates the participants' reflective activities, we asked the participants to complete a questionnaire to describe what they felt was learned

through constructing cognitive modeling. More concretely, the participants were given an instruction, "describe what was learned through creating cognitive models for the classes," and were required to answer the instruction in the questionnaire.

Class activities

The production system used in the class practice was identical to DoCoPro used in Class A and Class B.

- Introduction: In the class practice, intended initially for learning the basics of production system modeling, a model for performing an addition problem was constructed using the learning material.
- Simple problem solving: Subsequently the main learning phases followed. First they constructed a basic model for solving a simple subtraction problem, and then created a more complex model for solving a subtraction problem in which carry information must be processed to achieve a solution because the upper digit is smaller than the lower digit in a certain column.
- Complex problem solving: Finally, they were given a more difficult problem in which a digit in the upper row in a certain column was zero therefore carry information had to be repeated from the leftmost column. The following is an example problem.

```
  20000
 −10001
```

Results

The following is the analysis of the questionnaires answered by the 18 participants. The

contents of the answers were grouped into five categories.

(a) Reflective thinking on their own cognitive processing (9 among 18)

- Example 1: I understood how I usually perform addition and subtraction problems in my mind. I am very surprised that we can describe information processing in our brain accurately by using this production system.
- Example 2: Until now I have not been conscious of the ways I solve subtraction problems in my mind. Through constructing cognitive models, I have become aware of the procedures I use for calculation.

(b) Finding of complex and sophisticated information processing behind simple problem solving (8 among 18)

- Example 3: Usually we perform addition and subtraction problems without serious effort. I have noticed, however, that we do so thorough many complex steps in a process while drawing needed information from our memory.
- Example 4: What I learned by creating actual cognitive models is that even when solving simple calculation problems, information processing is performed in our mind through unexpectedly difficult processes.

(c) Noticing of the superiority of human cognitive processing and the limitations of production system models. (9 among 18)

- Example 5: Production system models basically can use only one fixed strategy. But humans can flexibly select one of multiple strategies according to the context.
- Example 6: This is the most essential difference between production system processing and our information procedures. … The fact that perception is

behind every act of cognitive processing is entirely a new finding discovered by using the production system in this class.

(d) Pointing out essential differences between human and machine information processing, expressing negative impressions about the cognitive modeling approach. (4 among 18)

- Example 7: I feel that what we do in our minds when solving problem is neither theoretic nor systematic. I think that humans behave more intuitively.

(e) Discovering new innovative information processing that is different from the ordinary procedures performed in dairy life. (2 out of 18)

- Example 8: I tried to find another way of solving the problem of modeling that was different than my usual way of doing so while thinking about various models. … As a result, I was able to find an innovative way of solving the problem. I think that this new way of finding a solution comes from the experience of understanding visually the process of thinking by using information processing models.

The descriptions categorized in (a) and (b) show evidence that the participants actually performed reflective activities while monitoring their own internal information processing. The descriptions in (c) imply that they learned the nature of human information processing more deeply through such activities. The above results indicate that most participants experienced such learning processes. On the other hand, there were some, but not so many, participants who gave impressions that we had not expected. The result in (e) is an unexpected but interesting result, and is discussed in detail later.

Discussion of Practice 2

There are three levels of learning in the learning from cognitive modeling: learning of programming, understandings of a learning domain, and findings on human cognitive processing. The first and second levels of learning, i.e., learning of programming and understanding of a learning domain, were supported by the log analysis and the posttest as reported in Practise 1. In the following, we discuss in detail the third level of learning based on the questionnaire.

The result of the questionnaire showed that most participants learned to reflect on their own cognitive information processing, and more generally discovered the superiority of human intellectual mental processing based on their experiences of cognitive modeling. As mentioned in the introduction section, such meta-cognitive activities activate learning, and there are a variety of trials to activate the learning processes by utilizing such meta-cognitive activities (Chi et al. 1989, 1994; Renkl 1997; Renkl et al. 1998; Pirolli and Recker 1994; Aleven and Koedinger 2002; Conati and Vanlehn 2000). The above results imply that constructing cognitive models has the possibility of directly activating such meta-cognitive activities.

Most participants encoded the procedures for subtraction that they had learned in their elementary school. The procedures include complex rules such as "if the upper number is less than the lower number in a processing column, shift attention to the left column, decrement the upper number in the focused-on column by one, and add ten to the upper number in the processing column," and "if the upper number in a focused-on column equals 0, then shift attention to the next left column, decrement the upper number in the focused-on column by one, shift attention back to the right, and add ten to the upper number in the focused-on column. However, some of the participants constructing the subtraction problems discovered a new procedure for subtraction while

constructing cognitive models. In the questionnaire, two participants actually mentioned such a new finding. The new procedure is as follows.

If

A column is processed

The answer slot in the column is empty (i.e., the answer has not been obtained in the column)

The digit in the upper row is smaller than the digit in the lower row in the column (i.e., impossible to perform the subtraction in the column)

Then

Add ten to the digit in the upper row in the current column

Add one to the digit in the lower row in the left column.

This procedure is not taught in Japanese schools; actually no participants knew this procedure before they participated in the class. To solve the example problem including zeros in the upper rows presented above, multiple complex rules are needed. On the other hand, the above procedure is very simple and sophisticated. If this single procedure is added to the basic rule set, the model can completely solve such kind of advanced problems. We should note that from the viewpoint of elementary education where acquiring the concept of the decimal system is a principal issue of calculation, there may be disadvantages in using this procedure. Anyway, this finding is interesting because discovering this procedure was brought about by the situation where the participants are forced to describe the calculation procedures formally based on the production system description. In the introduction section, we pointed out three

advantages of cognitive modeling. This finding may relate to "serendipity and emergence" as the third advantage.

Conclusions

The authors developed a web based production system for novice learners called DoCoPro. Using the system, we designed a class practice based on LtIP. In the final stage of the class practice, almost all participants successfully created relatively complex sophisticated models.

This result implies that LtIP functioned well as a learning design for teaching cognitive modeling. The log analysis showed that the participants learned the skills of programming for cognitive modeling using DoCoPro. They tried to monitor and reflect on their thinking processes to identify the cognitive processing taking place in their own minds; from these activities they learned some important aspects of human information processing. The participants who created more sophisticated models successfully understood the problem structures in the learning domain. However whether this advantage comes from the learning activity or their initial knowledge is unclear; to test this point is the most important future work.

The limited improvement of the models in Class A was observed. Our interaction design, LtIP, tells the participants the incompleteness of their models, and gives cues and criteria for the revisions, but does not give knowledge for improving the models. The participants had to find ways of model revision. Some liberal arts undergraduates without advanced programming skills noticed problems with their models but might not have known how to improve their models because of the lack of programming skills. Our interaction design was not intended to teach the participants programming skills directly; such programming skills were taught prior to the

interaction phase. To let LtIP function better for naïve students who are not knowledgeable about programming, we need to develop systematic ways to combine LtIP and such a programming learning environment.

It is important to distinguish learning by creating cognitive models from learning of creating models. The latter is important for cognitive science major students, but not for non-major and naïve students. We intended, by the phrase learning *by* creating cognitive models, that we use cognitive modeling as a tool to promote learning. Creating cognitive models is not the objective but a means for learning. Acquiring programming skills is very important for learning *of* cognitive modeling. A log analysis supported the development of the participants' programming abilities. However, this improvement is not sufficient for learning *by* creating models. To verify the utility of cognitive modeling as a learning tool, we also confirmed whether the participants successfully experienced monitoring their own mental processing, learned psychological processing, and understood domain knowledge. A posttest for an understanding of domain knowledge in Practice 1 and a questionnaire for learning of psychological mental processing in Practice 2 confirmed our expectations, supporting the possibility of learning by creating cognitive models.

Last let us summarize the novelty of LtIP as an instructional design of CSCL. LtIP provides a learning environment for implicit and asynchronous interaction among participants in which negative feedback that tells the incompleteness of each model makes differences of the models clear. A problem posed by each participant is a key factor, because it works as a mediator for causing such interaction. In LtIP, participants are naturally guided to generate as original problems as possible in a competitive situation of class activities. They also have a responsibility as inventors of original problems by checking if their own computational models can solve their proposed

problems. LtIP guarantees such originality and integrality in problem posing, resulting the successful class practices.

References

Aleven, V., & Koedinger, K. R. (2002). An effective metacognitive strategy: learning by doing and explaining with a computer-based Cognitive Tutor. Cognitive Science, 26, 147-179.

Anderson, J. R., & Lebiere, C. (1998). The atomic components of thought. Lawrence Erlbaum Associates, Hillsdale, NJ.

Aronson, E., & Patnoe, S. (1996). The jigsaw classroom: Building cooperation in the classroom. Addison Wesley Longman, NY.

Biswas, G., Leelawong, K., Schwartz, D., & Vye, N. (2005). Learning by Teaching: A New Agent Paradigm for Educational Software. Applied Artificial Intelligence, 19, 363-392.

Biswas, G., Jeong, H., Kinnebrew, J., Sulcer, B., & Roscoe, R. (2010). Measuring Self-regulated Learning Skills through Social Interactions in a Teachable Agent Environment. Research and Practice in Technology Enhanced Learning 5, 123-152.

Brown, J. S., & Burton, R. R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. Cognitive Science, 2, 155-192.

Chi, M. T. H., Feltovich, P. J., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. Cognitive Science, 5, 121-152.

Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. Cognitive Science, 13, 145-182.

Chi, M. T. H., Leeuw, de N., Chiu, M. H., & LaVancher, C. (1994). Eliciting self-explanations improves understanding. Cognitive Science, 18, 439-477.

Collins, T. D., & Fung, P. (2002). A visual programming approach for teaching cognitive modeling. Computers & Education, 39, 1-18.

Conati, C., & VanLehn, K. (2000). Toward computer-based support of meta-cognitive skills: A computational framework to coach self-explanation. Int. J. of Artificial Intelligence in Education, 11, 389-415.

Cooper, R. P., Fox, J., Farringdon, J., & Shallice, T. (1996). A systematic methodology for cognitive modeling. Artificial Intelligence, 85, 3-44.

Cooper, R. P. (2002). Modeling High-Level Cognitive Processes. Lawrence Erlbaum Associates, Mahwah, NJ.

Darden, L. (1992). Strategies for anomaly resolution. In R. N. Giere (Ed.), Cognitive models of science. Minnesota studies in the philosophy of science. Minneapolis: University of Minnesota Press.

Dourish, P., & Bellotti, V. (1992). Awareness and coordination in shared workspaces. Proceedings of computer human interaction 92, 541-548.

Dunbar, K. (2001). What scientific thinking reveals about the nature of cognition. In C. Crowley et al. (Eds.), Designing for science: Implications from everyday, classroom, and professional settings. Mahwah, NJ: LEA.

Ellerton, N. F. (1986). Children's Made up Mathematics Problems: A New Perspective on Talented Mathematicians. Educational Studies in Mathematics, 17, 261-271.

English, L. D. (1997). Promoting a Problem-posing Classroom. Teaching Children Mathematics, 4, 172-179.

Ericsson, K. A., & Simon, H. A. (1980). Verbal reports as data. Psychological Review, 87, 215-251.

Fidas, C., Komis, V., Tzanavaris, S., Avouris, N. (2005). Heterogeneity of learning material in synchronous computer-supported collaborative modeling. Computers & Education, 44, 135-154.

Fum, D., Missier, F. D., & Stocco, A. (2007). The cognitive modeling of human behavior: Why a model is (sometimes) better than 10,000 words. Cognitive Systems Research, 8, 135-142.

Gotel, Olly, Christelle Scharff, and Andrew Wildenberg. (2008). Teaching software quality assurance by encouraging student contributions to an open source web-based system for the assessment of programming assignments. Proceedings of the 13th annual conference on Innovation and technology in computer science education, 214-218.

Gros, B. (2001). Instructional design for computer-supported collaborative learning in primary and secondary education. Computers in Human Behaviour 17, 439–451.

Hansen, T., et al. (1999). Using telematics for collaborative knowledge construction. in P. Dillenbourg ed., Collaborative learning. Cognitive and computational approaches. Amsterdam: Pergamon.

Hirashima, T, et al. (2011). Learning by Problem-Posing for Reverse-Thinking Problems, Proceedings of the 15th international conference on artificial intelligence in education, 123-130.

Hirashima, T, et al. (2008). An Experimental Use of Learning Environment for Problem-Posing as Sentence-Integration in Arithmetical Word Problems, Proceedings of the 9th international conference on Intelligent Tutoring Systems, pp. 689-689.

Isotani, S., Inaba, A., Ikeda, M, & Mizoguchi, R. (2009). An ontology engineering approach to the realization of theory-driven group formation. Int'l Journal of Computer-Supported Collaborative Learning, 4, 445-478.

Klayman, J., & Ha, Y. (1987). Confirmation, disconfirmation and information in hypothesis testing. Psychological review, 94, 211-228.

Koedinger, K. R., Aleven, V., Heffernan, N., McLaren, B., & Hockenberry, M. (2004). Opening the door to non-programmers: authoring intelligent tutor behavior by demonstration. Proceedings of Seventh International Conference on Intelligent Tutoring Systems, pp. 162-174.

Kojima, K., & Miwa, K. (2008). A System that Facilitates Diverse Thinking in Problem Posing. International Journal of Artificial Intelligence in Education, 18, pp. 209-236.

Kojima, K., Miwa, K, & Matsui, T. (2010). Experimental Study for Design of Computational Learning Support to Enhance Problem Posing. Proceedings of 18th international conference on computers in education, 92-94.

Kulkarni, D., & Simon, H. A. (1988). The process of scientific discovery: The strategy of experimentation, Cognitive Science, 13, 139–176.

Larkin, J., & Simon, H. (1987). Why a diagram is (sometimes) worth ten thousand words. Cognitive Science, 11, 65–100.

Miwa, K. (2004). Collaborative discovery in a simple reasoning task, Cognitive Systems Research, 5, 41-62.

Miwa, K., Nakaike, R., Morita J., & Terai, H. (2009). Development of Production System for Anywhere and Class Practice. Proceedings of the 14th International Conference of Artificial Intelligence in Education, pp. 91-99.

Miyake, N., & Shirouzu, H. (2006). A collaborative approach to teaching cognitive science to undergraduates: The learning sciences as a means to study and enhance college student learning. Psychologia, 49, 101-113.

Morita J., Miwa, K., Nakaike, R., & Terai, H. (2009). Log Analysis of Outside Class Study for Cognitive Modeling. Proceedings of 17th international conference on computers in education, 346-350.

Nakaike, R., Miwa, K., Morita J., & Terai, H. (2011). Development of a Web-based Production System for Introductory Cognitive Science Classes. Transactions of the Japanese Society for Artificial Intelligence, 26, 536-546. (in Japanese)

Nakaike, R., Miwa, K., Morita J., & Terai, H. (2009). Development and Evaluation of a Web-based Production System for Learning Anywhere. Proceedings of 17th international conference on computers in education, 127-131.

Newell, A., & Simon, H. (1972). Human problem solving. Prentice-Hall, Englewood

Cliffs, NJ.

Newell, A. (1990). Unified theory of cognition. Harvard University Press, Cambridge, MA.

Ogata, H., & Yano, Y. (2000). Combining knowledge awareness and information filtering in an open-ended collaborative learning environment. Int'l Journal of Artificial Intelligence in Education, 11, 33-46.

Pew, R. W., & Mavor, A. S. eds. (2007). Human-system integration in the system development process: A new look. National Academy Press, Washington, DC.

Renkl, A. (1997). Learning from worked-out examples: a study on individual differences. Cognitive Science, 21, 1-29.

Renkl, A., Stark, R., Gruber, R.S.H., & Mandl, H. (1998). Learning from worked-out examples: The effects of example variability and elicited self-explanations. Contemporary Educational Psychology, 23, 90-108.

Ritter, F., Haynes, S., Cohen, M., Howes, A., John, B., Best, B., et al. (2006). High-level behavior representation languages revisited, Proceedings of seventh international conference on cognitive modeling, 404-407.

Ritter, F. (2009). Two cognitive modeling frontiers. Transactions of the Japanese society for Artificial Intelligence, 24, 241-249.

Schunn, C. D., Crowley, K., & Okada, T. (1998). The growth of multidisciplinary in the Cognitive Science Society. Cognitive Science, 22, 107-130.

Schwartz, D. L., Chase, C., Chin, D. B., Oppezzo, M., Kwong, H., Okita, S., Biswas, G.,

Roscoe, R., Jeong, H., & Wagster, J. (2009). Interactive Metacognition: Monitoring and Regulating a Teachable Agent. In D. J. Hacker, J. Dunlosky, and A. C. Graesser (Eds.), Handbook of Metacognition in Education.

Sewart, T. C. (2004). Teaching computational modeling to non-computer scientists. Proceedings of the sixth international conference on cognitive modeling, 386-387.

Silver, E. A., & Cai, J (1996). An Analysis of Arithmetic Problem Posing by Middle School Students. Journal for Research in Mathematics Education, 27, 521-539.

Strijbos, J. W., Martens, R. L., & Jochems, W. M. G. (2004). Designing for interaction: Six steps to designing computer-supported group-based learning. Computers & Education, 42, 403-424.

Yost, G. R. (1993). Acquiring knowledge in Soar, IEEE Expert, 8, 26-34.

Young, R., & O'Shea, T. (1983). Errors in children's subtraction. Cognitive Science, 5, 153-177.