Sequential Structure of Objective Knowledge with an Application to Learning System

Yamasaki, Susumu; Sasakura, Mariko

***Abstract*—**We deal with knowledge structure of (operation) sequences, where the operation may contain information on acquisition of objective knowledge. A sequence of operations makes a performance of procedure causing situation transitions, where the system based on formation of operation sequences has characters such that: (1) the operations are regarded as media for objective knowledge, which may denote not only objects but also primitive procedures, (2) the situation is referred to by name, and (3) the organization of operation sequences may be automated. We apply such knowledge structure to implementation of a learning system such that it is mainly constructed to take formation of operation sequences for exercise practice in programming language.

Index Terms—*Knowledge structure*

1. INTRODUCTION

This paper deals with a model for a method of how to organize an automated process of learning. The conceived theory is concerned with a system for sequential structure of objective knowledge. As regards the proposed system, some artificial intelligence frameworks may be relevant.

(1) There have been some trend in e-learning systems, regarding adaptive aspects [9][19] or the tutoring one [22].

(2) The logical analysis standpoint has contained a wide range of formal systems since its organization ([11][15][17]).
- Hybrid logic, which involves both state-dependent and modal operators, is a formal system with logical meanings of states and worlds ([1][2]).
- Relations between the events are discussed through predicates in classical and modal logic ([7][13]). The event as the cause-and-effect relationship is made clear from the view of complexity ([10]).

(3) Correlation between action and knowledge has been also studied ([20]). A mathematical behaviour of action is also formulated in [18].

(4) The agent technology style is compiled in [21], where algebraic approach originates from [12][16].

As current topics, the e-learning methodologies are closely related to the problem which this paper deals with. Among the methodologies, the e-learning systems involve adaptive aspects [8] so that the exercise may be devoted to adaptation. From the visualization view of points, we have an implemented system of adaptive e-learning ([24]). The concept of adaptation is regarded as primary for e-learnings as in [4][5][6]. On the other hand, the concept of tutoring for navigation as in [3] is relevant to methodologies of self-learning. Observing adaptation and tutoring methodologies, we pay attention to mechanization of learning process with reference to exercise practice.

To learn programming languages, the exercise practice can be of use. This is because some compilation of exercise practice to acquire knowledge may conceive automated process of learning step by step:

- An exercise is by itself basic, if we complete it.
- An exercise leads to subsidiary exercises for recovery learning, if we do not complete the original exercise.

We can continue to learn by ourselves, trying exercises and following the way depending on whether we complete the given exercise, or we do not complete but take recovery with subsidiary exercises. If we could observe an automated process of learning as above mentioned, then it may be mechanized for the learner's manual. This is why we organize an automated process of learning with reference to exercise practice.

We focus on the design problem of a formal system for exercise practice in programming language. The formal system contains the constraints:

Manuscript received June 18, 2008.
Susumu Yamasaki is with Department of Computer Science, Okayama University, Japan
(e-mail: yamasaki@momo.cs.okayama-u.ac.jp).

(a) the exercises are organized in advance.
(b) the practices are interactive with respect to situations.
(c) an interactive effect of practice causes situation transition and a step to the next exercise.

We then design a formal system whose mechanism is analyzed as follows. A sequence of operations $x_1 \cdots x_n \ (n \geq 0)$ makes a performance of the form $\sigma_1, Sem[x_1], \sigma_2, Sem[x_2], \sigma_3, \cdots, \sigma_n, Sem[x_n], \sigma_{n+1}$ where $Sem[x_i] \ (1 \leq i \leq n)$ denotes an implementation of the operation x_i, and $\sigma_1, \cdots, \sigma_{n+1}$ are a corresponding sequence of situations. The system regarding formation of operation sequences has characters such that

(i) the operations contain objective knowledge,
(ii) the situation is referred to by name, and
(iii) the organization of operation sequences may be automated.

The paper is organized as follows. Section 2 is concerned with a formal system. In Section 3, we have a procedure for formation of operation-sequences. In Section 4, we have an application of the system to exercise practices in programming. Section 5 contains concluding remarks.

2. FORMAL SYSTEM FOR KNOWLEDGE STRUCTURE

For the formation of operation sequences, we formulate a formal calculus of illustrating it in [23][28]. Different from the previous works, the system of this paper is constructed with top down rule base so that it is applied to exercise practices in programming. Compared with the former version, we have elaborate points for implementation of a language learning system. If we need a rule to substitute a sequence of operations for the operation x, that is, a logic program: $x \leftarrow$, or $x \leftarrow y_1, \cdots, y_n \ (n > 0)$, then the performance may be
$Sem[\varepsilon], Sem[x],$
$or\ Sem[y_1], \cdots, Sem[y_n], Sem[x].$
In this paper, to make the rule of substituting operations for an operation by top-down design, we regard it as defined for an operation x such that $x \to \varepsilon$, or $x \to y_1, \cdots, y_n \ (n > 0)$, and the performance is a sequence of
$Sem[x], Sem[y_1], \cdots, Sem[y_n],$
$or\ Sem[x], Sem[\varepsilon].$
That is,

The sequential relation of operations is in this paper determined by rewriting rules, but not by logic programs with negation (as in [25][27][29][30]).
The top-down application of operations is preferable, while the bottom-up application is adopted in the former version.
A performance is caused by implemented operation sequences with situation transition sequences:
$\sigma_1, Sem[x_1], \sigma_2, Sem[x_2], \sigma_3, \cdots, \sigma_n, Sem[x_n], \sigma_{n+1}$
.
A rewriting rule is used as a form: $x \to \varepsilon$ (empty sequence), or $x \to y_1, \cdots, y_n \ (n > 0)$ for an operation x to be involved in a sequence of performance. In this section, the system is formulated by means of rewriting rules.
This version of the system is a model to effectively perform a sequence of operations for learning.
A system is a quadruple $\Im = (C, \sum, Sem, R)$, where:

(i) C is a set of operations.
(ii) $\sum$ is a set of situations.
(iii) $Sem : C \to (\sum \to \sum)$ is a semantic function.
(iv) R is a set of rewriting rules of the form $A \to \alpha \ in \ C \times C^*$. Note that $C^* = \{x_1 \cdots x_n \mid n \geq 0, x_1, \cdots, x_n \in C\}$.
The empty sequence in C^* is denoted by ε.

A member of C^* is a sequence of operations. The semantic function Sem is extended. The original function assigns a situation transition to each operation. Intuitively speaking, the extended function Sem assigns a situation transition to each sequence of operations so that it gives a meaning of a sequence over objective knowledge of operation.

Definition 1. The semantic function Sem is extended to be a function $Sem : C^* \to (\sum \to \sum)$ by:

(1) $Sem[\varepsilon]\sigma = \sigma$.
(2) $Sem[\gamma x]\sigma = Sem[x](Sem[\gamma]\sigma)$ $(x \in C, \gamma \in C^*)$.

Inference rules for $\Im$ by means of the follower relation R :

We define the derivation as the least set satisfying the closure of following inference rules (1), (2) and (3), on the assumption that a system $\Im = (C, \sum, Sem, R)$ is given. (See [14] for such formality in signed data.) We denote the

derivation of $move_R(G;\sigma_1;\sigma_2)$ by applying the inference rules (1)-(3) finitely many times, with the predicate $move_R(G;\sigma_1;\sigma_2)$.

(1) $\dfrac{}{move_R(\varepsilon;\sigma;\sigma)}$

(2) $\dfrac{(x \to G) \in R \quad move_R(G;\sigma_3;\sigma_2) \quad Sem[x]\sigma_1 = \sigma_3}{move_R(x;\sigma_1;\sigma_2)}$

(3) $\dfrac{move_R(G_1;\sigma_1;\sigma_4) \quad move_R(G_2;\sigma_4;\sigma_2)}{move_R(G_1G_2;\sigma_1;\sigma_2)}$

In other words, the relation $move_R \subseteq C^* \times \sum \times \sum$ is defined such that by $move_R(\gamma;\sigma_1;\sigma_2)$, we mean that:
Given the sequence γ initiated, the situation transition from σ_1 to σ_2 is caused by rewriting and reducing γ to the empty sequence.

3. FORMATION OF OPERATION SEQUENCES

Semantics of a sequence of operations is defined for the system $\Im = (C, \sum, Sem, R)$. The following lemma suggests a relation of the application of the Sem function to concatenation of two sequences β and γ with the composition of Sem functions.

Lemma 1. Assume a system $\Im = (C, \sum, Sem, R)$. Then
$Sem[\beta\gamma]\sigma = Sem[\gamma](Sem[\beta]\sigma)$.

Proof. It is proved by induction on the structure of the sequence γ.

(1) In the case that $\gamma = \varepsilon$,
$$Sem[\beta\gamma]\sigma = Sem[\beta]\sigma = Sem[\gamma](Sem[\beta]\sigma)$$

(2) In the case that $\gamma = \gamma_1 x$ for some $x \in C$,
$$\begin{aligned} Sem[\beta\lambda]\sigma &= Sem[\beta\gamma_1 x]\sigma \\ &= Sem[x](Sem[\beta\gamma_1]\sigma) \\ &= Sem[x](Sem[\gamma_1](Sem[\beta]\sigma)) \\ &\quad (by\ induction\ hypothesis) \\ &= Sem[\gamma_1 x](Sem[\beta]\sigma) \\ &= Sem[\gamma](Sem[\beta]\sigma) \end{aligned}$$

Now assume a system $\Im = (C, \sum, Sem, R)$. We need a procedure to form a sequence of operations for the system $\Im$. The procedure contains not only deterministic cases but also nondeterministic ones to get an existing sequence. ***Operation-sequence formation for*** $\Im$***:***

$Formation(G;\sigma_1;\sigma_2) \Leftarrow$
if $G = \varepsilon$
then
  if $\sigma_1 = \sigma_2$ **then** ε (empty sequence)
else
  if $G = xG_2$ such that $x \to G_2$ is in R
  then
    if $Sem[x]\sigma_1 = \sigma_3$
    then $x.Formation(G_1G_2;\sigma_3;\sigma_2)$
  else
    if $G = G_1G_2$ such that
      $Formation(G_1;\sigma_1;\sigma_4)$ and
      $Formation(G_2;\sigma_4;\sigma_2)$ are defined
    then
$Formation(G_1;\sigma_1;\sigma_4)Formation(G_2;\sigma_4;\sigma_2)$

By means of the above procedure $Formation$, we can have an operation-sequence formation for the relation $move_R(G;\sigma_1;\sigma_2)$. That is, if we could have the predicate $move_R$ on the triplet of G (a sequence of operations), and two situations σ_1 and σ_2, we may have some sequence β to cause the situation transition from σ_1 to σ_2.

Theorem 1. Assume that $move_R(G;\sigma_1;\sigma_2)$ for some $\sigma_1, \sigma_2 \in \sum$. It follows that
$\exists \beta \in C^*.[Sem[\beta]\sigma_1 = \sigma_2]$.

Proof. Assume that $move_R(G;\sigma_1;\sigma_2)$ for some $\sigma_1, \sigma_2 \in \sum$. Take the procedure for operation sequences. By structural induction on the sequence, we see that if $Formation(G;\sigma_1;\sigma_2)$ yields γ, then $Sem[\gamma]\sigma_1 = \sigma_2$.

(1) If $G = \varepsilon$, then $\sigma_1 = \sigma_2$ so that $Formation(\varepsilon;\sigma_1;\sigma_1) = \varepsilon$ and $Sem[\varepsilon]\sigma_1 = \sigma_1$.

(2) If $G = xG_2$ for some operation x and $Sem[x]\sigma_1 = \sigma_3$ such that $move_R(xG_2;\sigma_1;\sigma_2)$, then we must assume that some rule $x \to G_1$ is in R, and $move_R(G_1G_2;\sigma_3;\sigma_2)$. Assume that $Formation(G_1G_2;\sigma_3;\sigma_2)$ yields γ_1, and that $Sem[\gamma_1]\sigma_3 = \sigma_2$. By the procedure, $Formation(G;\sigma_1;\sigma_2)$ provides $x\gamma_1$. It follows that

$$Sem\left[x\gamma_1\right]\sigma_1 = Sem\left[\gamma_1\right]\left(Sem\left[x\right]\sigma_1\right) = Sem\left[\gamma_1\right]\sigma_3 = \sigma_2$$

(3) If $G = G_1G_2$ such that relations $move_R\left(G_1;\sigma_1;\sigma_4\right)$ and $move_R\left(G_2;\sigma_4;\sigma_2\right)$, then we must assume that $move_R\left(G_1G_2;\sigma_4;\sigma_2\right)$. Assume that $Formation\left(G_2;\sigma_1;\sigma_4\right)$ yields a sequence β and $Formation\left(G_2;\sigma_4;\sigma_2\right)$ yields a sequence γ, respectively such that $Sem\left[\beta\right]\sigma_1 = \sigma_4$ and $Sem\left[\gamma\right]\sigma_4 = \sigma_2$. It follows that $Formation\left(G_1G_2\right)$ provides a sequence $\beta\gamma$ such that

$$Sem\left[\beta\gamma\right]\sigma_1 = Sem\left[\gamma\right]\left(Sem\left[\beta\right]\sigma_1\right) = Sem\left[\gamma\right]\sigma_4 = \sigma_2$$

by means of Lemma 1. This concludes the proof.

Implementation for Exercise Practice

For a learner to practice exercise, the system is implemented. The system is regarded as providing a learning course. That is, automated process of learning is offered by means of courses involving exercises in the programming language ML (as in [26]).

Module	*Elements*
Database	*Operation table* *Exercise table* *Course table* *(A set of rewriting rules)*
Operation management	*Operation provider* *Grade checker* *(Generation of semantic function)* *Next operation decider* *(Based on the inference rule)*

Table 1. System Construction.

The system is constructed as shown in Table 1, where it contains two modules.

Databse which is constructed by means of MySQL, for data involved in the system.

Operation management (engine) which provides chart sequences with interaction of learners.

(1) Database

The system makes use of three tables:

Operation table

Exercise table

Course table

These tables are supposedly given by some expert in ML programming.

(a) The *Operation table* involves the operations which learners use. The operation contains the explanations and examples. It is observed by Web-browser such that it is written in terms of html-file. The *Operation table* consists of a relation between the operation and its html-file.

(b) The Exercise table involves examination problems for test, by which learners' grade of understanding is evaluated. The examination problem is made by some expert and observable by Web-browser.

(c) The *Course table* involves courses of learning languages from operations in order. They are rule-based.

(2) Operation management engine

The *Operation management* is a module to provide the operation, evaluate the answer of the learner for the examination problem, and decide what operation to be next provided. It contains three parts:

Operation provider which demonstrates the indicated operation for each course.

Checking the grade of understanding for learners.

Decision of the next operation which is made for the course.

For example of an ML learning course, we have an operation named by "*ML_standard_course*" which contains operations named by:

(a) "*BasicType*"
(b) "*Function Definitions*"
(c) "*Local Environment*"
(d) "*Exception*"
(e) "*HigherOrderFunctions*''
(f) in the case that we could not be successful in some interaction regarding the operation *ML_standard_course*.

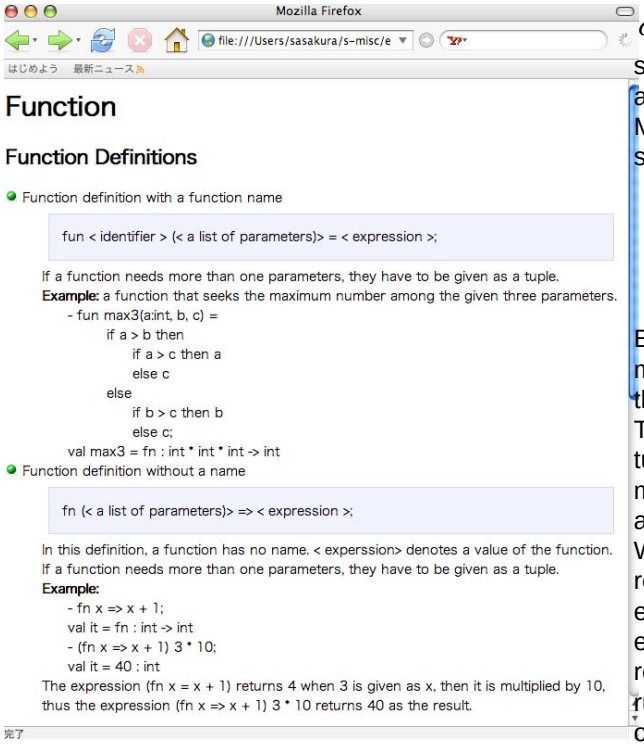

Figure 1. Web page for definition.

Even if a routine of the process fails, some recovery routine is ready until it is successful. That is, the rewriting system is adopted like the forms:

$x \to \varepsilon$ (in a successful case)

$x \to y_1 \cdots y_n$ (in a recovery case)

We demonstrate a part for exercise practices of some subsets of ML. We present a structure of function definition containing an operation "*FunctionDefinitions*" in Fig. 1. It contains syntactic explanations in the cases of function definitions with name and without name. In each case, a simple and typical example is illustrated.

By the operation, we can refer to the exercise. Three questions and exercises are set, which are based on [26], while the system may work in response to answers for the exercises. The reaction of the system causes a situation transition so that the system, by rewriting rules, provides the next operation(s).

4. CONCLUSION

We present a formal system to obtain a performance of sequences

$\sigma_1, Sem[x_1], \sigma_2, Sem[x_2], \sigma_3, \cdots, \sigma_n, Sem[x_n], \sigma_{n+1}$

where $Sem[x_i]$ $(1 \le i \le n)$ denotes an implementation of the operation x_i, and $\sigma_1, \cdots, \sigma_{n+1}$ are a corresponding sequence of situations. Based on the system, we implement an e-learning system for the beginner to practice ML programming exercise. We are now ready to say that:

(i) The operation sequence is a learning process in terms of rewriting rules.

(ii) The situation is abstract to be applied to some constraint on the formation of operation sequences.

Because the rewriting rule is regarded as a given method for self-learning, this paper suggests that this formal system is applicable to an e-learning. The methodology is not always adaptive, nor a tutoring. The implemented system offers a method different from proper adaptive methods and tutoring systems.

We have some remark on the manner of giving rewriting rules. For the system to implement exercise practices, we must design: (a) the exercise assigned to each operation, (b) the relation of the operations by means of rewriting rules, and (c) the situation transition which an operation causes.

We may assume the integrity constraint on the situation set, if the situation transition should be restricted. As far as the integrity constraint on the set of situations is recursively enumerable, we can effectively form a sequence of operations for the specified initial sequence with the situation transition.

REFERENCES

[1] Areces,C. and Blackburn,P., "Repairing the interpolation in quantified logic", Annals of Pure and Applied Logic, 123, pp. 287-299, 2003.

[2] Brauner,T., "Natural deduction for hybrid logics", J. of Logic and Computation, 14, pp. 329--353, 2004.

[3] Brusilovsky,P., Schwarz,E. and Weber,G., "ELM-ART: an intelligent tutoring system on World Wide Web Intelligent Tutoring Systems", Lecture Notes in Computer Science, 1086 (Proceedings of 3rd International Conference on Intelligent Tutoring Systems, ITS-96), pp.261--269, 1996.

[4] Brusilovsky,P. and Nijhavan,H., "A framework for adaptive e-learning based on distributed re-usable learning activities", Proceedings of World Conference on E-learning 2002, pp.154-161, 2002.

[5] Brusilovsky,P. and Maybury,M.T., "From adaptive hypermedia to the adaptive web", Communications of ACM, 45, 5, pp.31-33, 2002.

[6] Brusilovsky,P., "KnowledgeTree: a distributed architecture for adaptive E-Learning",

Proceedings of 13th international World Wide Web conference, pp.104-113, 2004.
[7] Cervesato,I., Chittaro,L. and Montanari,A., “A general modal framework for the event calculus and its skeptical and credulous variants”, Proc. of 12th European Conference on Artificial Intelligence, pp.12-16, 1996.
[8] Conejo,R., Guzm\'{a}n,E., Mill\'{a}n,E., Trella,M., P\'{e}rez-De-La-Cruz,J.L. and R\'{i}os,A., “ SIETTE: a web-based tool for adaptive testing”, International Journal of Artificial Intelligence in Education, 14, pp.29-61, 2004.
[9] Conlan,O., Wade,V., Bruen,C. and Gargan,W., “Multi-model, metadata driven approach to adaptive hypermedia services for personalized e-learning, Adaptive Hypermedia and Adaptive Web-Based Systems”, Second International Conference, AH 2002, pp.100-111, 2002.
[10] Dean,T. and Boddy,M., “Reasoning about partially ordered events”, Artificial Intelligence, 36, pp.375-399, 1988.
[11] Genesereth,M.R. and Nilsson,N.J., “Logical Foundations of Artificial Intelligence”, Morgan Kaufmann, 1988.
[12] Hoare,C.A.R., “Communicating Sequential Processes”, Prentice-Hall, 1985.
[13] Kowalski,R.A., “Database updates in the event calculus”, J. of Logic Programming, 12, pp.121-146, 1992.
[14] Kunen,K., “Signed data dependencies in logic programming”, J. of Logic Programming, 7, pp. 231-245, 1989.
[15] Lloyd,J.W., “Foundations of Logic Programming”, 2nd, Extended Edition,Springer-Verlag, 1993.
[16] Milner,R., “Communication and Concurrency”, Prentice-Hall, 1989.
[17] Minker,J. (ed.), “Foundations of Deductive Databases and Logic Programming”, Morgan Kaufmann Publishers, Inc., 1987.
[18] Mosses,P.M., “Action Semantics”, Cambridge University, 1992.
[19] Paramythis A. and Loidl-Reisinger S., “Adaptive learning environments and e-learning standards”, Electronic Journal on e-Learning, 2, 1, pp.181-194, 2004
[20] Reiter,R., “Knowledge in Action”, The MIT Press, 2001.
[21] Russell,S. and Norvig,P., “Artificial Intelligence -A Modern Approach-”, Prentice-Hall, 1995.
[22] Ritter,S., Anderson,J., Cytrynowicz,M. and Medvedeva,O., “Authoring content in the PAT algebraic tutor”, Journal of Interactive Media in Education, 98, 9, pp.1-30, 1998.
[23] Sasakura,M., Iwata,K. and Yamasaki,S., “An interactive environment for generating sequential information”, Proc. of 10th International Conference on Information Visualization IV06, pp.441-446, London, 2006.
[24] Sasakura,M. and Yamasaki,S., “A framework for adaptive e-learning systems in higher education with information visualization”, Proc. of 11th International Conference on Information Visualization IV07, pp.819-824, Zurich, 2007.
[25] Shepherdson,J.C., “Negation in logic programming”, In Minker,J. (ed.), Foundations of Deductive Databases and Logic Programming, pp. 19-88, 1987.
[26] Ullman,J.D., “Element of ML programming”, Prentice Hall International, Inc., 1994.
[27] Yamasaki,S. and Sasakura,M., “An automated reasoning for diagnostic knowledge in a distributed environment”, Proc. of International Symposium on Information and Communication Technologies ISICT03, pp.547-552, Dublin, 2003.
[28] Yamasaki,S. and Sasakura,M., “A calculus effectively performing event formation with visualization”, Proc. of ISHPC-VI, LNCS. 4759, pp.287-294, 2008.
[29] Yamasaki,S., “Logic programming with default, weak and strict negations”, Theory and Practice of Logic Programming, 6, pp. 737-749, 2006.
[30] You,J.-H. and Yuan,L.Y., “On the equivalence of semantics for normal logic programs”, J. Logic Programming, 22, pp. 211-222, 1995.

Susumu Yamasaki received Eng.D. in information science from Kyoto University in 1980. In 1985/86, he was a visiting fellow at Dept. of Computer Science, University of Warwick, U.K. In 1987, he joined Okayama University as a professor.

Mariko Sasakura received Eng.D. in information science from Kyushu University in 2000. She is an assistant professor of Okayama University.