

# A GRAPH MANIPULATION SYSTEM ABSTRACTED FROM E-LEARNING

Susumu Yamasaki and Mariko Sasakura

Department of Computer Science, Okayama University, 3-1-1 Tsushima-Naka, Okayama, Japan

Keywords: Graph manipulation, e-Learning Abstraction, Knowledge engineering.

Abstract: In this position paper, we have an outlook on a graph manipulation system applicable to a visual interface of managing educational courses for an e-Learning system. We paraphrase a process of learning into a scheme of state-transitions related to graph manipulations. By the scheme, techniques and tools for educational courses can be abstracted to the treatments of graphs. Thus we construct a methodology of graph manipulations for a management of educational courses by extending the established graph viewer. We also propose a simple sketch-based interface to manipulate graphs. We here list up several tools of the sketch-based interface, based on the algorithm to detect *mouse movements* for indicating operations of the system.

## 1 INTRODUCTION

To develop an e-Learning system, there are various problems, one of which is concerned with creating and making contents of e-Learning. For contents-making, we are required to care specifications of teachers, where contents-making contains (i) a design of education courses, (ii) making education materials and (iii) a course management.

An *adaptive* management of courses includes interesting aspects of knowledge engineering, since it is just to manage contents of courses with respect to learners' ideas (which may be interactive with teachers). So far a purpose of this position paper is to show a method regarding the course managements.

As regards the materials to be made for an education course, an abstraction from adaptive e-Learning systems may be reasonable, because: (a) The material induces a function from texts to an abstract operation (which is an idea or a notion for education). An abstract operation is related to others under a situation, where the situation is an abstract state of a learner with respect to a given (computing) environment. (b) From the views of concrete versus abstract aspects, concrete objects as means are required, while abstract ones fitting thoughtful manners of users are demanded, with reference to the notion of situation.

For such an abstraction, we conceive a state-transition model. We then have problems for course managements: (i) How can we construct a course? (ii) What kind of representation is needed as a construc-

tion of courses? (iii) How can we represent objects as course materials? (iv) What formulation of course management with human interface can we make?

To have an insight into such problems, a graph knowledge structure (which may reflect the state-transition model) is available such that its construction may be implementable techniques. In Section 2, we present a state-transition model (graph manipulation) regarding self-learning systems. In Section 3, graph manipulation tools are described, to some extent, with reference to managing educational courses. Section 4 briefly suggests a graph manipulation system (Sasakura and Yamasaki, 2008) and some primary results of this paper, compared with relevant works.

## 2 ABSTRACTION FROM E-LEARNING SYSTEM

### Adaptive e-Learning Systems

E-Learning systems have become important in higher education, especially in universities. Many commercial products or open source systems have been developed, such as Blackboard, WebCT, or Moodle. Generally speaking, e-Learning systems may consist of several features like remote education, assistance for communication between a teacher and students with e-mails, or bulletin board systems. We have discussed self-learning systems in which students read materials

and take examinations by themselves whenever they want. Teachers provide materials and exercises in advance, and check results of students at intervals.

An adaptive e-Learning system (Brusilovski et al., 2004) is one of self-learning systems. It generates an educational course in accordance with understanding of a student. Since a student can review materials which he/she does not master at an appropriate time, the decrease of his/her motivation will be prevented. The problem in adaptive e-Learning systems is the way how to build a course that suits to each student. Knolmayer (2003) used a decision making method to build a course. Marshall and Metchell (2004) proposed to use the Capability Maturity Model and SPICE that are results of research of software engineering. Conlan et al. propose a qualitative model for adaptive e-Learning (Colan et al., 2002). Conejo et al. have made a tool to generate exercises automatically (Conejo et al., 2004). Brusilovsky and Maybury (2002) have proposed to use the Web system to build an adaptive system.

On the backgrounds of adaptive e-Learning systems, we see that contents-making of an e-Learning system consists of three aspects: (i) The first one is to design an educational course. (ii) The second one is to make educational materials. (iii) The third one is to arrange the materials in an educational course.

The problems of contents-making are classified into: (1) making educational materials (w.r.t. the second aspect), and (2) arranging materials in educational courses (w.r.t. the third aspect).

With respect to making educational materials, the problem is how teachers could make ‘good’ materials. About the arrangement of materials, the problem is how teachers could handle educational materials and construct educational courses. In this paper, an educational course is regarded as a list of educational materials which a student may learn, following the instructions. Because the managing problem may be related to computing researches, we here present the managing tools to assist teachers arranging educational courses.

#### A Logical Structure Applicable to Educational Courses

By means of a self-learning system, we may model operations and situations, where the operations are abstracted from educational materials, and situations represent statuses of how students understand. The situation may be changed, following what materials are learned by a student. We assume that the transition of the situation can be known by an object reflecting the material, such that we have a scheme  $\mathfrak{S} = (O, \Sigma, R, Int)$ , where: (i)  $O$  is a set of operations.

(ii)  $\Sigma$  is a set of situations. (iii)  $R \subseteq (\Sigma \times O) \times O$ . (iv)  $Int : L \times Env \rightarrow \Sigma$  is a mapping such that  $L$  is a set of learners and  $Env$  a set of Web-environments.

$O^*$  stands for a set of all finite operation sequences. A sequence of operations in  $O^*$  is possibly interpreted as a process of learning.

An interaction is implemented by the function  $Int$ , where we abstract such a function, assuming that a learner in an (computing) environment has a situation. On the other hand, a pair of a situation and an operation is supposedly related to another operation, by means of the relation  $R$ . Such formal definitions represent the behaviour to virtually display an interactive human interface for the course working of the e-Learning system which we are now concerned with.

A relation  $Seq \subseteq O \times O$  is defined:

$$Seq(o_1, o_2) \Leftrightarrow \begin{aligned} & \text{(i) } o_1 = o_2, \text{ or} \\ & \text{(ii) } \exists \sigma_1, \dots, \sigma_{m+1} \in \Sigma, \exists o'_1, \dots, o'_m \in O. \\ & R((\sigma_1, o_1), o'_1), R((\sigma_2, o'_1), o'_2), \dots, \\ & R((\sigma_m, o'_{m-1}), o'_m), R((\sigma_{m+1}, o'_m), o_2). \end{aligned}$$

That is,  $Seq(o_1, o_2)$  means that there is a sequence of operations beginning with  $o_1$  and ending with  $o_2$ .

Owing to solvability of reachability in graph theory, we can have:

*Proposition:* Given a scheme with a finite set of  $Env$  and for any learner, it is solvable whether  $Seq(o_1, o_2)$  holds for any given two operations  $o_1$  and  $o_2$ .

As we may be aware of a relation between the above proposition and graph reachability, this model may be translated to a graph manipulation, with which the basis of the proposed system is described in Section 4.

### 3 GRAPH MANIPULATIONS VIRTUALLY MANAGING COURSES

#### Operations for Managing Courses

Considering a close relation between the state-transition model (as in the previous section) and the graph structure, we design graph operations, abstracted from the management of educational courses. A graph consists of nodes and edges. At first, we list up the operations which are needed to create and manipulate a graph, based on existing graph description languages and graph drawing tools (Adar, 2006).

We have the operations of three groups: (1) the operations for changing a structure of a graph, (2) the

operations to change a view of a graph, and (3) the operations to manipulate a graph drawing system: The operations of the first group change a logical structure of a graph, the second group's operations do not change a logical structure but change an occurrence of a graph, while the third group's operations do not change a graph.

The operations included in the first group can change the structure of a graph: To create/remove a node, to create/remove an edge, and to change an edge's start point or end point.

The second group includes the operations which change the characteristics of a node or an edge. We point out seven operations: To move a node, to change size/color/line style of a node, to add/change a label of a node, to change colour/style of an edge, to add/change a label of an edge, to change style/size of an arrow head of an edge, and to gather edges.

The third group includes four operations which do not change anything of a graph, but provide useful functions of a graph drawing system: To scale up/down, to undo/redo, to save/load, and grouping/ungrouping nodes and/or edges.

Our motivation comes from educational courses. To keep it in mind and to construct a manipulation package, we would examine a separation of the manipulation stage for educational courses from the creating one for educational courses.

For a creation of educational courses, teachers must make the contents. In a mechanized system, "creating a node" requires to create new contents for the node. So far, "creating a node" operation and relative operations are omitted here from our graph manipulating system. We will provide another interface to create a node.

As a result, the operations of the present system are (a) the operations to change a structure of a graph except "creating a node", and (b) the operations for a system except "grouping/ungrouping" and "changing views".




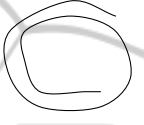

Therefore, we design an interactive system which provides 8 operations: (1) Moving a node, (2) Moving an edge (Changing an edge's start or end point), (3) Deleting a node, (4) Deleting an edge, (5) Adding an edge, (6) Undoing, (7) Redoing, and (8) Scaling up.

A graph is loaded, when the system is invoked. Saving a graph is performed automatically while the system is running.

#### A Visual Representation of the Operations

For selected 8 operations of manipulating educational courses, we can design a simpler system without complicated menus and buttons. Instead of menus or buttons, we detect special mouse movements of 8 operations.

Table 1: Visual commands of the system.

Representations	Names	Operations
•	Click	Scaling up
	Line	Moving
	Crossing lines	Deleting
	Arrow	Drawing
	CCW circle	Undoing
	CW circle	Redoing

tions.

As the "mouse"-device movements, we can detect: click, drawing a line, drawing a crossing lines, drawing an arrow, drawing a circle by clockwise, and drawing a circle by counter-clockwise.

Interpreting the movements as the situation demands, we can implement the 8 operations in the present system. The visual representation of the operations are shown in Table 1.

We define *Element* which is a set of atoms (atomic visual representations). *Element* is

$$\{Click, Line, CCW\ circle, CW\ circle, Node, Edge\},$$

where *Click* is represented by a "dot", *CCWcircle* and *CWcircle* are circles with clockwise and counter-clockwise rotations, respectively. *Node* and *Edge* are elements of a graph.

We define the visual representations of our interface:  $Visual\ representation = Element^+$ , where  $Element^+$  is a nonempty set of elements constructed by at least one atom.

## 4 A GRAPH MANIPULATION SYSTEM

The system contains graph manipulations described in Section 3. We have developed a system which manipulates graphs for course management (Sasakura and Yamasaki, 2008).

We have presented a part graph of a course for the

programming language ML (Ullman, 1994). The layout of this graph is designed by the *dot* program which is a layout program for directed graphs. The nodes indicate educational materials. They are separated by 4 colors. At the same time, the lines are to point out some directions. The *arrows* indicate the order of learning. The *crossing red* lines are user's inputs which specify the deletion of a node. The *red CCW circle* specifies "Undoing" operation to the graph.

So far, we proposed a simple sketch-based interface for materials of an e-Learning system. Using the interface, we can select commands for managing a tree by surveying existence of graph editors and considering characteristics of materials of adaptive e-Learning systems. We then described how users specify the commands by mouse movements. Introducing two predicates which describe the relation of plural strokes of a mouse, we can specify a command as multi-stroke drawing. We also mentioned algorithms to implement the proposed simple sketch-based interface.

As a primary result of this position paper based on the system (Sasakura and Yamasaki, 2008), we have presented tools of graph manipulations by which a whole system has been developed with human interface. As above mentioned, we make an analysis on the e-Learning system in terms of (i) course designs, (ii) making education materials, and (iii) a course management. To organize a course management, we see the process of self-learning by means of state-transitions which can be represented by graph manipulations containing 8 operations as in Section 3. They are closely related to knowledge engineering, from the views of knowledge structure as well as ontology developments.

By the system of this paper, we have provided sketch-based interface not to create a graph but to manipulate graphs. Our contribution to graph manipulations would easily implement an intuitive interface for graph manipulations by making use of sketch-based interface.

Before our approaches, there have been relevant works, as we have before met: Herman et al. make a survey on many graph visualization and navigation techniques in information visualization (Herman et al., 2000). Freire and Rodriguez (2004) present a graph-based direct manipulation interface and maintain complex hypermedia structure with well-known graph drawings and visualization techniques. GUESS (Adar, 2006) is a system for graph exploration.

The interface used in our proposed system is one of sketch-based interfaces, while sketch-based interfaces are often used for drawing some diagrams, such as Unified Modeling Language (UML) diagrams

which are included in a popular visual modeling language for software engineering. Many UML editors with sketched-based interface have been proposed (Chen et al., 2008).

In early times, sketch-based interface systems used Rubine's algorithm (Rubine, 1991) which recognizes symbols drawn by a single stroke. Recently, to recognize complicated symbols, multi-stroke recognition systems were proposed (Hse et al., 2004).

## REFERENCES

- Adar, E. (2006). *GUESS: a language and interface for graph exploration. Proceedings of the SIGCHI conference on Human Factors in computing systems*, 791–800.
- Brusilovsky, P. (2004). KnowledgeTree: a distributed architecture for adaptive E-Learning. *Proceedings of the 13th international World Wide Web conference*, 104–113.
- Chen, Q., Grundy, J., Hosking, J. (2008). SUMLOW: early design-stage sketching of UML diagrams on an E-whiteboard. *Software Practice and Experience* 38, 961–994.
- Conejo, R., Guzman, E., Millan, E., Trella, M., Perez-De-La-Cruz, J. L. and Rios, A. (2004). *SIETTE: a web-based tool for adaptive testing. International Journal of Artificial Intelligence in Education* 14, 1–33.
- Conlan, O., Wade, V., Bruen, C. and Gargan, W. (2002). Multi-model, metadata driven approach to adaptive hypermedia services for personalized eLearning. *Adaptive Hypermedia and Adaptive Web-Based Systems: Second International Conference*, 100–111.
- Gansner, E. R. and North, S. C. (2000). An open graph visualization system and its applications to software engineering. *Software Practice and Experience* 30, 1203–1233.
- Herman, I., Melançon, G. and Marshall, M. S (2000). Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics* .6, 1, 24–43.
- Hse, H. and Newton, A. R. (2004). Sketched symbol recognition using Zernike moments. *Proceedings of the 2004 ACM Symposium on Pattern Recognition*, 367–370.
- Lara, J. and Vangheluwe, H. (2004). Defining visual notations and their manipulation through meta-modelling and graph transformation. *Journal of Visual Languages and Computing* 15, 309–330.
- Rubine D. (1991). Specifying gesture by examples. *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, 329–337.
- Sasakura, M. and Yamasaki, S. (2008) A graph manipulation visual interface for construction of e-Learning systems. *Proc. of 12th International Conference on Information Visualization*, 644–649.
- Ullman, J. D. (1994). Element of ML programming. *Prenice Hall International*.