Curriculum Knowledge Representation and Manipulation in Knowledge-Based Tutoring Systems

Gang Zhou, Jason T.-L. Wang, *Member, IEEE*,
and Peter A. Ng, *Member, IEEE Computer Society*

Abstract—A Knowledge-Based Tutoring System (KBTS) is a computer-based instructional system that uses artificial intelligence techniques to help people learn some subjects. We found that the knowledge communication process involving a KBTS and a human student can be decomposed into a series of communication cycles, where each cycle concentrates on one topic and contains four major phases: planning, discussing, evaluating and remedying. The major contributions of this work are the development of a generic architecture for supporting the knowledge communication between a KBTS and a student, and a graphical notation and schema for supporting the curriculum knowledge representation and manipulation during the planning phase of a tutoring process. The curriculum knowledge about a course can help a tutoring system determine the sequences in which the topics will be discussed with the students effectively and diagnose the students' mistakes. The curriculum knowledge base contains the goal structure of the course, prerequisite relations, and multiple ways of organizing topics, among others. As an example, we have focused on developing SQL–TUTOR, a KBTS for the domain of SQL programming. This system has features such as efficient control mechanism, explicit curriculum knowledge representation, and individualized private tutoring. For allowing the students relative freedom to decide how to study the domain knowledge about a subject, the system provides the students with a group of operators to hand-tailor the learning schedules according to their special backgrounds, requests, and interests.

Index Terms—Computer in education, intelligent tutoring, knowledge representation and manipulation, topic association graph, learning graph, curriculum management.

1 Introduction

"Knowledge-Based Tutoring Systems" (*KBTSs*) [10], [33] or "Intelligent Tutoring Systems" (*ITSs*) [1], [13], [24], [27], [29], [37], [39] use artificial intelligence techniques to help a person learn some subjects. The goal of KBTS research is to build private tutoring systems that can "understand" *what*, *when*, *how*, and *whom* they are teaching and can tailor their contents and methods to meet the needs of an individual learner without being limited to a repertoire of predefined responses [12], [31]. In [27], Reiser et al. reported that students working with private tutors can learn the given materials four times faster than those students who study in a classical classroom by attending lectures, reading texts, and working alone on homework problems. Bloom [2] found that students working with private tutors have a better grasp of the materials than a comparable group of students spending the same amount of time in the classroom.

According to Burger and Desoi [5], Marcenac [17], and Wenger [35], a KBTS usually consists of three major components:

1) domain knowledge base: keeping the subject materials and the skills that the system intends to teach to a student;
2) student model: representing a student's knowledge about the subject domain (i.e., what the student does and does not know); and
3) pedagogical knowledge base: describing the tutoring strategies of the system.

Many of the AI knowledge representation schemes, including semantic network [38], [39], logic program [19], qualitative model [36], [37], and production system [1], [27], have been successfully used by the existing KBTSs to represent the various kinds of knowledge in the systems.

However, this simple decomposition of a KBTS into three components does not provide a solid framework upon which a KBTS tailored to a specific domain can be developed effectively. When we started to build SQL–TUTOR [40], [41], a KBTS for tutoring SQL programming, we found that the current KBTS research suffers from two drawbacks. The first one is that the system control mechanism is not well defined. Because most of the KBTS research has been initiated primarily to explore the capability of AI techniques in the process of learning and teaching, the focus of the existing KBTS projects has been on the knowledge representation aspects of the components of the system (domain knowledge, student model, pedagogical knowledge, etc.), rather than on control features. Typically, most of the KBTSs are research prototypes that focus on

• *The authors are with the Institute for Integrated Systems Research, Department of Computer and Information Science, New Jersey Institute of Technology, Newark, NJ 07102.*
E-mail: gzhou@homer.njit.edu, jason@village.njit.edu, ngp@probe.njit.edu.

Manuscript received Jan. 13, 1995.
For information on obtaining reprints of this article, please send e-mail to: transkde@computer.org, and reference IEEECS Log Number K96057.

only one or two components in the systems; this obscures the need for coordination among the components. As a result, how to use the knowledge to accomplish a tutoring task is ill-defined and how to implement a domain-specific tutoring system is still a big burden for KBTS researchers.

The second drawback is that the representation of the curriculum knowledge has not been fully addressed, and there is little effort to formalize the curriculum knowledge representation and manipulation. The curriculum knowledge about a subject domain is a type of meta-level knowledge which describes the goal structure of the subject materials, the different perspectives to organize the subject materials, and the order in which the subject materials should be presented to and discussed with the students. Consequently, it plays an important role during the planning and evaluating phases of a communication cycle, and in improving the tutoring effectiveness. However, most of the current KBTSs do not contain this type of knowledge. A few systems encoded the curriculum knowledge in their knowledge bases, but it was encoded implicitly in the domain knowledge base [11], rather than a separate system component.

The intent of this paper is to overcome these drawbacks by providing a framework for constructing KBTSs effectively. The work makes two major contributions. First, we identify the underlying control flow inside a tutoring process and introduce our SQL–TUTOR architecture that supports this flow (Section 2). Second, we formalize the representation of the curriculum knowledge and develop a graphical notation and schema to support the manipulation of the knowledge during the planning phase of a communication cycle (Section 3 and Section 4). For exposition purposes, we illustrate our approach using examples drawn from our SQL–TUTOR prototype, though the techniques described here can be generalized to other KBTSs designed for tutoring different subjects.

2 A Generic Architecture for KBTSs

With a few exceptions [4], [6], [7], [22], [23], [26], most KBTS research has focused on domain knowledge representation [36], [37], [39], student model design [3], [30], [32], [33], [34], and pedagogical knowledge representation [9], [17], [18], as opposed to the procedures for controlling a system's behavior during a tutoring process. However, understanding the control is equally important because it determines:

1) What a tutoring system should be able to do in order to accomplish its tutoring task. A tutoring process is a very complicated procedure that usually includes several steps, such as selecting a proper topic for a student, answering questions raised by the student about the topic, evaluating the student's mastery of the topic, diagnosing and helping the student to correct his/her errors.
2) How to order these steps so that the system will do the right thing at the right time. For example, the system should know when it is appropriate to end the discussion of one topic and select a new topic to study.
3) When and where to retrieve what types of knowledge stored in the system's knowledge bases during a tutoring process. For example, after a topic is selected, the system will access the domain knowledge base to find the subject contents associated with the topic in order to present the contents to the student.

Let us now focus on the activities of a KBTS during a tutoring process and analyze what types of control procedures are necessary for a KBTS. Our goal is to discover what elements constitute an effective tutoring process and how they interact with each other. First, we explore a general tutoring process involving a human instructor and students in some detail. Then, we summarize the underlying control steps in this process. Finally, we introduce a generic architecture based on our analysis that supports this tutoring process.

In a class of 40–50 minutes, a good instructor never pours out the whole contents of a lesson to students without any intervention. Now and then, he/she observes carefully the students' reactions to his/her lectures by asking them questions and by encouraging them to raise their own questions. In case the students have difficulties in understanding his/her lecture, the instructor will stop the lecture and try to figure out why the students do not understand it and how to help them. The instructor will not start a new subject unless he/she is convinced that the students are indeed following what he/she is teaching.

To summarize, the instructor has divided a class into a series of periods and concentrates on only one topic per period. In each of these periods, the instructor goes through several phases: selecting the topic to be studied, discussing the materials related to the topic, getting feedback from the students, finding out their problems, and helping them when they have difficulties. We call such a teaching period a *communication cycle*, because this is a two way process: the instructor not only gives lectures to the students, but also uses the feedback from the students to guide his/her teaching. In other words, the instructor communicates with the students.

Similarly, the tutoring process in an environment involving a KBTS and a student can also be viewed as a series of communication (negotiation) cycles.[1] Each communication cycle consists of four phases: planning, discussing, evaluating, and remedying (see Fig. 1). The tasks of these phases can be specified as follows:

1) *Planning*: selecting, by either the student or the system, a new topic for the student to study.
2) *Discussing*: displaying to the student the contents of the selected topic stored in the system, and answering the student's questions about the selected topic.
3) *Evaluating*: posing problems associated with the selected topic to the student and evaluating his/her performance by analyzing his/her solutions.
4) *Remedying*: taking any necessary pedagogical actions to correct the student's errors found in his/her solutions.

1. Note that Moyse [20], Moyse and Elson-Cook [21], and Wenger [35] also viewed a tutoring process as knowledge communication, though these authors did not divide the process into cycles or divide each cycle into phases.

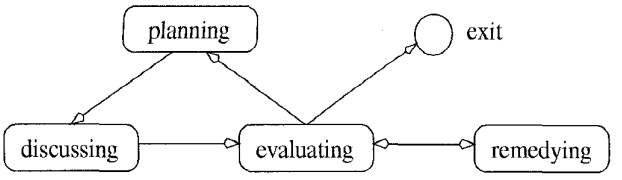


Fig. 1. A communication cycle in a KBTS; each cycle consists of four phases: planning, discussing, evaluating, and remedying.

In the design of SQL–TUTOR, we have developed a novel architecture that supports this view of tutoring SQL as knowledge communication. Besides a domain knowledge base (DKB), a student model (SM), and a pedagogical knowledge base (PKB), which often exist in other KBTSs, we have created two new system components, namely a curriculum knowledge base (CKB) and a communication controller (CC). The communication controller is a set of procedures organized into four modules: planning, discussing, evaluating, and remedying module. The behavior of the system during a tutoring process is determined by these four modules, as described below:

1) During the planning phase, the system can adopt either an *active* or a *passive* strategy. In an active planning mode, the student is in the control of the topic selection. He/she can choose any topic (or remove any optional topic) from the course. Based on the curriculum knowledge stored in the CKB, the system will suggest a set of schedules to the student for his/her study without forcing him/her to follow these schedules. It is up to the student to decide how to go through the course. In this mode, the student is an *actor*, who receives advice from the planner.

   In the active mode, after the student chooses a topic, the planning module makes reference to the CKB in order to generate a learning graph according to the student's interests. Each learning graph defines a set of schedules which guide his/her study of a subset of the course materials (the formal definition of the learning graph and its properties will be given in Section 4). The planning module traverses the generated learning graph, picks the topics contained in the graph in a certain order, and passes them, one at a time, to the discussing module.

   In a passive planning mode, the system is in control of the topic selection. It will select a proper topic to tutor the student by

   - accessing the student model to determine his/her current knowledge status, i.e., the knowns and unknowns about the domain;
   - consulting the CKB to find the prerequisite relationships among the student's unknowns and generate a topic hierarchy according to the relationships;
   - accessing the PKB to find out the tutoring strategies which determine the selection of the next topic and apply them to choose an appropriate topic;
   - finally, following the same procedure as we have described for the active planning mode, i.e., generating a learning graph for the selected topic and passing the topics contained in the learning graph to the next module, namely, the discussing module.

2) Once a topic is received from the planning module, the discussing module is in control. Its tasks include presenting the contents of the current topic to the student and answering the student's questions regarding this topic. (The student is allowed to raise questions through a menu-driven interface.) The following steps are taken to accomplish the tasks:

   - accessing the domain knowledge base to retrieve the contents (concepts and skills) relevant to the topic;
   - generating a lecture explaining the concepts or skills of the selected topic to the student;
   - analyzing questions raised by the student and creating internal representations for these questions;
   - reasoning about the domain knowledge stored in the DKB and trying to answer questions;
   - displaying the solutions to the student if they exist.

3) During the evaluating phase, the system's task is to measure a student's mastery of the current topic. The evaluating module accomplishes this task by

   - selecting a set of problems associated with the topic from the domain knowledge base;
   - converting the problems into a form that is understandable by the student, and presenting these problems to the student;
   - reading and analyzing the solution given by the student, and generating the result data from its analysis which indicate whether the solution is correct, partially correct, or totally wrong;
   - updating the student model to reflect whether the student has mastered the material perfectly, or is making some progress during his/her learning process;
   - diagnosing the student's missing conceptions or misconceptions if his/her performance is not satisfactory. Whenever an error is found, the system passes it to the remedying module.

4) During the remedying phase, the remedying module helps the student fix his/her errors found in the evaluating phase. The possible remedial actions include:

   i) giving the student an example, hint, a partial solution, or a complete solution;
   ii) letting the student read the text again and concentrate on some specific parts;
   iii) leading the student to identify his/her errors by asking some questions.

   The remedying strategies stored in the PKB define these remedying actions.

The architecture of SQL–TUTOR is shown in Fig. 2. It has the following features:

1) Providing a generic framework for KBTS construction which can be applied to various domains;
2) Incorporating the view that a tutoring process consists of a series of communication cycles, and each cycle consists of four phases and focuses on one specific topic;

TABLE 1
SOME TOPICS IN SQL–TUTOR AND THEIR CONCEPT SETS

TOPIC NAME	CONCEPT SET
CONDITIONAL RETRIEVAL	{WHERE CLAUSE, SEARCH CONDITION}
SIMPLE RETRIEVAL	{}
SIMPLE SEARCH EXPRESSION	{SIMPLE COMPARISON, RELATIONAL OPERATOR, PRECEDENCE RULE-1}
ONE TABLE SIMPLE RETRIEVAL	{COLUMN LIST, RESULT TABLE}
COMPOUND RETRIEVAL	{}
COMPOUND SEARCH EXPRESSION	{LOGICAL OPERATOR, PRECEDENCE RULE-2}
ONE TABLE COMPOUND RETRIEVAL	{}
MULTI-TABLE COMPOUND RETRIEVAL	{JOIN}

Note that the PRECEDENCE RULE *in* SIMPLE SEARCH EXPRESSION *is different from the* PRECEDENCE RULE *in* COMPOUND SEARCH EXPRESSION. *The former describes the precedence rules among relational operators, whereas the latter describes the precedence rules among logical operators. In each concept set, only a small collection of concepts are shown here.*

3) Clarifying the underlying control mechanism for a tutoring process and showing how a tutoring process is accomplished effectively;
4) Clarifying the roles played by the knowledge bases and the interactions among the control procedures.

Thus, SQL–TUTOR is designed to fully support the four phases in the communication cycles. Detailed discussions of the four phases are beyond the scope of this paper. Instead, in the following sections, we will focus on the design of the curriculum knowledge base and its role in the planning phase.

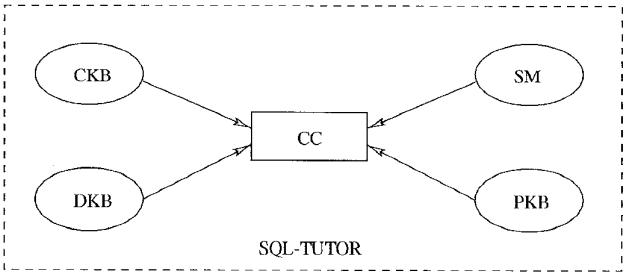


Fig. 2. The SQL–TUTOR architecture.

3 CURRICULUM KNOWLEDGE REPRESENTATION

In this section, we formalize the description of the curriculum knowledge in a tutoring system. We define the *domain knowledge* of a tutoring system TS, denoted as $DK(TS)$, as a set of domain concepts.

DEFINITION 1. *A topic T in a tutoring system TS is a bi-tuple $T = (N_T, C_T)$, where N_T and C_T are the name and the concept set of the topic, respectively. The concept set is a subset of the domain knowledge of TS, that is, $C_T \subset DK(TS)$.*

A topic in a tutoring system may correspond to a part, a chapter, a section, a subsection, etc., of a textbook, whose content consists of a set of domain concepts. Table 1 shows some topics in SQL–TUTOR and their concept sets.

An important feature about the topics in a tutoring system is that they are not isolated. Instead, they are related to one another in various ways. The relationships among the topics may have a great impact on the effectiveness of the system. Therefore, it is necessary for a tutoring system to formulate the knowledge concerning the relationships among the topics. We call this kind of knowledge the *curriculum knowledge* of the system.

We have identified three typical topic relations: *subtopic-of*, *view-of*, and *prerequisite-of*. In the following subsections, we discuss in detail these relations and their representations in a tutoring system.

3.1 *Subtopic-Of* Relation

A topic T in a tutoring system can usually be decomposed into smaller topics, which are often called the *subtopics* of that topic. A subtopic, in turn, can be decomposed into even smaller topics. This process of decomposing the topic T into subtopics will eventually generate a tree structure with each node associated with a topic, and the children of the node associated with its subtopics. We call such a tree a *topic tree* of T, denoted as $\mathcal{T}(T)$. Fig. 3 shows a part of the topic tree for topic CONDITIONAL RETRIEVAL in Table 1.

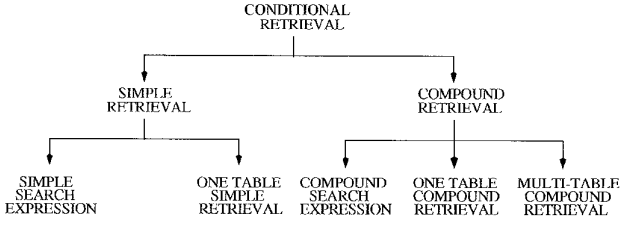


Fig. 3. A partial topic tree of CONDITIONAL RETRIEVAL. Each node is labeled by a topic's name with its concept set omitted.

A leaf topic $(N_{T'}, C_{T'})$, $C_{T'} \neq \emptyset$, in $\mathcal{T}(T)$ is also called a *unit topic* (or *unit*). Topic T_1 is a *subtopic* of topic T_2, denoted as $\mathbf{s}(T_1, T_2)$, if T_1 is a descendant of T_2 in $\mathcal{T}(T)$. Topic T_1 is a *child subtopic* of T_2, denoted as $\mathbf{cs}(T_1, T_2)$, if T_1 is a child of T_2 in $\mathcal{T}(T)$.

DEFINITION 2. *Let T be a topic of a tutoring system TS. The domain of T in TS, denoted as $Dom(T)$, is defined as follows:*

1) *if $T = (N_T, C_T)$ is a unit topic, then the domain of T is its concept set, that is, $Dom(T) = C_T$;*
2) *if $T = (N_T, C_T)$ is not a unit topic, then its domain is the union of its concept set and the domains of its child subtopics, that is, $Dom(T) = C_T \cup Dom(T_1) \cup \ldots \cup Dom(T_n)$, where $\mathbf{cs}(T_i, T)$ and $1 \leq i \leq n$.*

Finally, the domain of a topic tree is the domain of its root.

Note that if $T_1 = (N_{T_1}, C_{T_1}), \ldots, T_p = (N_{T_p}, C_{T_p})$ represent all the subtopics of topic $T = (N_T, C_T)$, the domain of T can also

TABLE 2
SOME TOPICS IN SQL–TUTOR AND THEIR DOMAINS

TOPIC	DOMAIN
CONDITIONAL RETRIEVAL	{WHERE CLAUSE, SEARCH CONDITION, SIMPLE COMPARISON, RELATIONAL OPERATOR, PRECEDENCE RULE-1, COLUMN LIST, RESULT TABLE, LOGICAL OPERATOR, PRECEDENCE RULE-2, JOIN}
COMPOUND RETRIEVAL	{LOGICAL OPERATOR, PRECEDENCE RULE-2, JOIN}
COMPOUND SEARCH EXPRESSION	{LOGICAL OPERATOR, PRECEDENCE RULE-2}
ONE TABLE COMPOUND RETRIEVAL	{}
MULTI-TABLE COMPOUND RETRIEVAL	{JOIN}

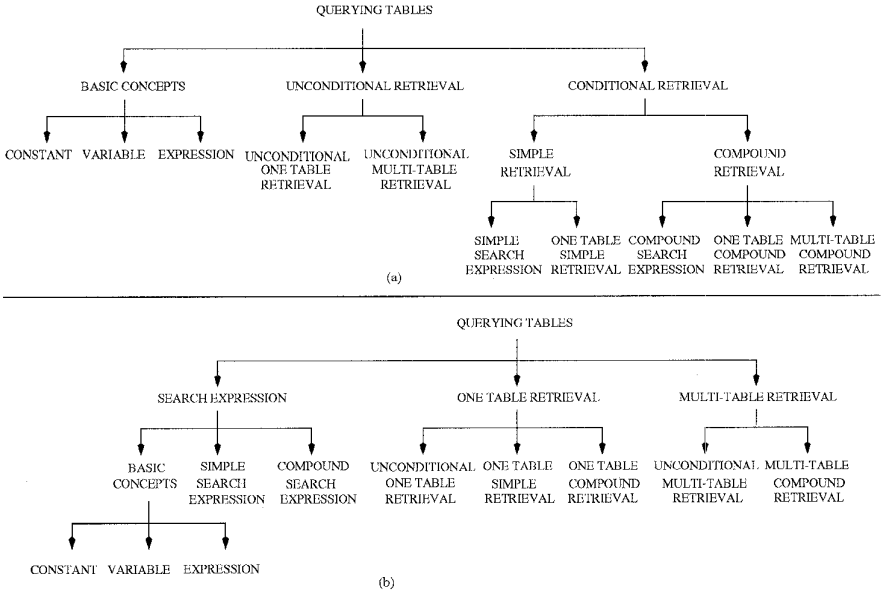


Fig. 4. Two topic trees of **QUERYING TABLES**.

be written as $Dom(T) = C_T \cup C_{T_1} \ldots \cup C_{T_p}$. Table 2 lists the domains of **CONDITIONAL RETRIEVAL** and some of its subtopics (cf. Fig. 3 and Table 1).

DEFINITION 3. *A topic tree $\mathcal{T}(T)$ is* well defined *if for any two distinguished topics, $T_1 = (N_{T_1}, C_{T_1})$ and $T_2 = (N_{T_2}, C_{T_2})$ of $\mathcal{T}(T)$, we have*

1) $C_{T_1} \cap C_{T_2} = \emptyset$; *and*
2) $Dom(T_1) \neq Dom(T_2)$.

It is easy to prove that if topics T_1 and T_2 are not ancestors of each other in a well defined topic tree, then their domains are disjoint (i.e., $Dom(T_1) \cap Dom(T_2) = \emptyset$).

3.2 *View-Of* Relation

For the given domain in which a tutoring system is designed, there are usually several ways to organize the domain concepts into topic trees based on various factors. As an example, Fig. 4 shows parts of two topic trees of **QUERYING TABLES**. For exposition purpose, we include only some of the topics here. In general, there can be much more topics in the trees. We now explore some possible relationships among these topic trees.

DEFINITION 4. *Let $\mathcal{T}_1(T_1)$ and $\mathcal{T}_2(T_2)$ be two topic trees. $\mathcal{T}_1(T_1)$ and $\mathcal{T}_2(T_2)$ are* compatible *if $Dom(T_1) = Dom(T_2)$.*

Different compatible topic trees reflect different approaches to organize the domain knowledge in a tutoring system. The capability of organizing the material of a topic by different ways allows a system to use different approaches to teach a student to achieve the set of teaching goals associated with the same topic. There are two major advantages to allow a tutoring system to incorporate several topic trees in its curriculum knowledge base:

1) Certain organizations are more effective for some students to study the topic than others. It is desirable that an instructor can always select the best organization for each student. By the same token, a tutoring system providing multiple topic trees over its domain concepts can improve the teaching effectiveness by adopting different approaches, based on individual needs of the students, to teach different students over the same topic.
2) The system can provide different organizations to teach a student the same concepts in different situations. For example, when a student studies a topic for the first time, the system can use one organization to teach the topic. Later on, the student can select a different organization to review this topic. Therefore, the student can study the same topic from the different perspectives and has a better understanding of the content of the topic.

We call each way to organize the materials of a topic a *view* of the topic. Formally, a view can be defined as follows:

DEFINITION 5. *A* view *of topic T is a well defined topic tree of T.*

Topic T_1 is said to be a subtopic of topic T_2 with respect to a view $\mathcal{T}(T)$, denoted as $\mathbf{s}(T_1, T_2, \mathcal{T}(T))$, if T_1 is a subtopic of T_2 in $\mathcal{T}(T)$. When the context is clear, we still use $\mathbf{s}(T_1, T_2)$ to denote that T_1 is a subtopic of T_2, regardless of the view under which this relation holds.

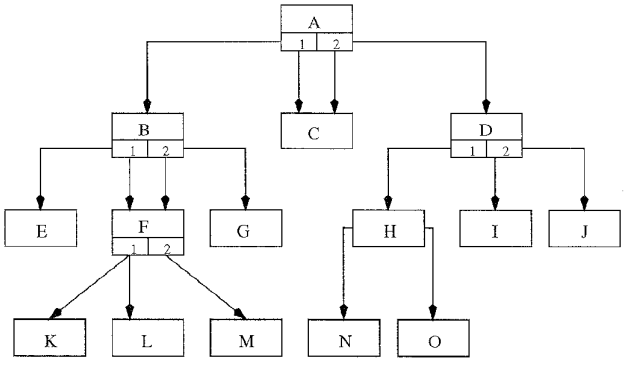


Fig. 5. Illustration of multiple views of topics (nodes); numerical labels within a node are used to distinguish the multiple views of the node.

Fig. 5 shows how multiple views can be associated with topics. In this figure, topic F has two views. The first view consists of topics K and L, whereas the second view consists of topic M. Numerical labels within a node are used to distinguish the multiple views of the topic. Therefore, the two views of F are written as F(7) = {K, L} and F(8) = {M}, respectively. The following lists the views of topics A, B, and D:

```
A(1-3-7) = {A, B, C, E, F, K, L}
A(1-3-8) = {A, B, C, E, F, M}
A(1-4-7) = {A, B, C, F, G, K, L}
A(1-4-8) = {A, B, C, F, G, M}
A(2-5) = {A, C, D, H, N, O}
A(2-6) = {A, C, D, I, J}
B(3-7) = {B, E, F, K, L}
B(3-8) = {B, E, F, M}
B(4-7) = {B, F, G, K, L}
B(4-8) = {B, F, G, M}
D(5) = {D, H, N, O}
D(6) = {D, I, J}
```

3.3 Topic Association Graph and *Precedence-Of* Relation

In addition to the topic-subtopic and view relationships, there is another kind of curriculum relationship, the precedence relationship, among domain concepts and topics. This relationship can be used by the system to determine i) the order in which topics and concepts to be selected and presented to the students during a tutoring process, and ii) the missing prerequisite knowledge if the student can not find the correct answer to a problem. Roughly speaking, concept C_1 is a *precedence of* concept C_2 if C_1 is used to define (describe, or explain) C_2 and therefore, C_1 should be taught before C_2. We use $\mathbf{p}(C_1,C_2)$ to denote that concept C_1 is a precedence of concept C_2.

By its nature, the precedence relation among the domain concepts forms a partial order (i.e., it is irreflexive and transitive). Based on this relation, we can define the precedence relation among a set of topics as follows:

DEFINITION 6. *Let* $T_1 = (N_{T_1}, C_{T_1})$ *and* $T_2 = (N_{T_2}, C_{T_2})$ *be two distinguished topics in a view* $\mathcal{T}(T)$ *of a tutoring system.* T_1 *is a* precedence of (*or prerequisite of*) T_2 *in* $\mathcal{T}(T)$, *denoted as* $\mathbf{p}(T_1, T_2, \mathcal{T}(T))$ (*or simply* $\mathbf{p}(T_1, T_2)$ *if the context is clear*), *if one of the following is true:*

1) *there are two domain concepts,* C_1 *and* C_2, *such that* $C_1 \in C_{T_1}, C_2 \in C_{T_2}$ *and* $\mathbf{p}(C_1, C_2)$ (first order precedence);
2) T_1 *and* T_2 *are not ancestors of each other in* $\mathcal{T}(T)$ *and there is a topic* T_3 *such that* $\mathbf{s}(T_3, T_1, \mathcal{T}(T))$ *and* $\mathbf{p}(T_3, T_2, \mathcal{T}(T))$;
3) T_1 *and* T_2 *are not ancestors of each other in* $\mathcal{T}(T)$ *and there is a topic* T_3 *such that* $\mathbf{s}(T_3, T_2, \mathcal{T}(T))$ *and* $\mathbf{p}(T_1, T_3, \mathcal{T}(T))$.

Consider again Table 1 and Fig. 3. If both concepts SIMPLE COMPARISON and RELATIONAL OPERATOR are precedences of concept LOGICAL OPERATOR, we obtain the following precedence relations:

- p(SIMPLE SEARCH, EXPRESSION, COMPOUND SEARCH EXPRESSION) (first order precedence).
- p(SIMPLE RETRIEVAL, COMPOUND SEARCH EXPRESSION) (Item 2 of Definition 6).
- p(SIMPLE RETRIEVAL, COMPOUND RETRIEVAL) (Item 3 of Definition 6).

If we combine the precedence relations among topics and the various *views* of topics, we obtain a Topic Association Graph (*TAG*). Formally, a *TAG* can be defined as follows:

DEFINITION 7. *A* Topic Association Graph *in a tutoring system is a quadruple* $TAG = (\mathcal{N}_s, \mathcal{N}_c, \mathcal{E}_s, \mathcal{E}_p)$, *where*

- $\mathcal{N}_s$ *is the set of nodes associated with only one view (simple nodes);*
- $\mathcal{N}_c$ *is the set of nodes associated with more than one view (complex nodes);*
- $\mathcal{E}_s$ *is the set of topic-subtopic relations (edges); and*
- $\mathcal{E}_p$ *is the set of precedence relations.*

Fig. 6 illustrates a portion of the *TAG* in SQL–TUTOR's curriculum knowledge base in which the subtopic-of and the first order precedence-of relations among topics are represented by the solid and dotted edges, respectively. There are two views associated with QUERYING TABLES: The first view consists of nodes QUERYING TABLES, BASIC CONCEPTS, UNCONDITIONAL RETRIEVAL, CONDITIONAL RETRIEVAL, and their subtopics, whereas the second view consists of nodes QUERYING TABLES, SEARCH EXPRESSION, ONE TABLE RETRIEVAL, MULTI-TABLE RETRIEVAL, and their subtopics. In this example, QUERYING TABLES is the only complex node, though in general, any node in a *TAG* can have multiple views as shown in Fig. 5.

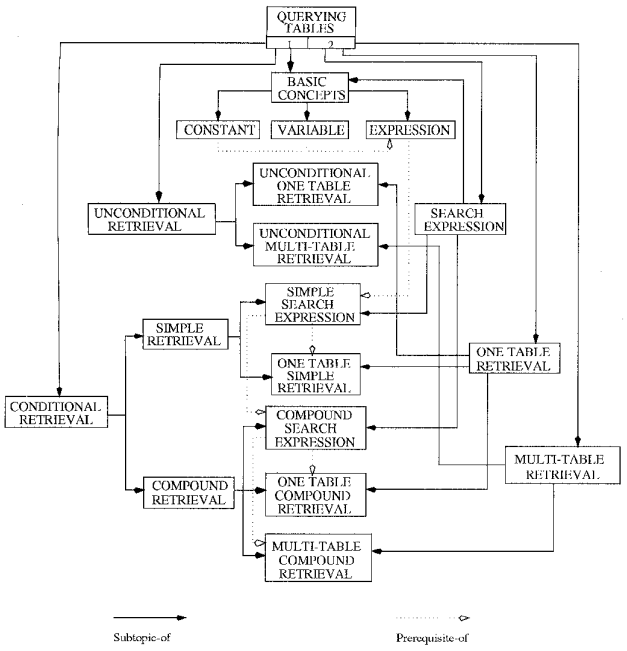


Fig. 6. A portion of the topic association graph (*TAG*) in SQL–TUTOR (curriculum knowledge about querying tables).

Let T_1, T_2, T_3, and T_4 be distinguished topics in view $\mathcal{T}(T)$ in a *TAG*. The following propositions hold:[2]

PROPOSITION 1 (Transitivity). *If* $\mathbf{p}(T_1, T_2)$ *and* $\mathbf{p}(T_2, T_3)$, *then* $\mathbf{p}(T_1, T_3)$.

PROPOSITION 2. *Topic* T_1 *is a precedence of* T_2 *iff there exist two domain concepts,* C_1 *and* C_2, *such that* $C_1 \in Dom(T_1)$, $C_2 \in Dom(T_2)$, *and* $\mathbf{p}(C_1, C_2)$.

PROPOSITION 3. *If* $\mathbf{s}(T_1, T_2)$, $\mathbf{s}(T_3, T_4)$, *and* $\mathbf{p}(T_2, T_4)$, *then* $\mathbf{p}(T_1, T_3)$.

4 Curriculum Knowledge Manipulation

In this section, we discuss how to manipulate the curriculum knowledge about a course represented by a *TAG* so that private tutoring can be conducted effectively.

4.1 Private Tutoring

One of the most important features of private tutoring is that it is sensitive to the following characteristics of each individual student:

1) the student's knowledge states (i.e., his/her knowns and unknowns) about the subject materials prior to and during the tutoring; and
2) the student's special learning needs, requests, and interests concerning the subject materials.

There are two possible approaches that KBTSs can use to conduct private tutoring. In the first approach, the system takes a directive role in controlling the tutoring from the beginning to the end. Throughout the whole process of interaction, the student can raise questions, but nothing else. Most of the existing KBTSs adopt this approach. They maintain two knowledge bases: the student model and the pedagogical knowledge base. The student model contains information about the students' understanding and mastery of the domain subjects. This information allows a KBTS to tutor a student based on his/her background and performance on the subjects, and to use different teaching procedures and materials to teach different students.

The pedagogical knowledge base contains the knowledge of teaching strategies. A KBTS can adopt different teaching strategies during a tutoring process. The teaching strategies can be chosen based upon:

- the students' knowledge states maintained in the student model (for instance, a system can give more examples and explanations to a novice programmer than to an experienced programmer);
- the types of topics (for example, a system can choose the coaching strategy while teaching a student problem solving skills and choose Socratic strategy while tutoring domain concepts); and
- the stages of problem solving (for example, a KBTS may give a very simple hint if a student has difficulty to answer a question the first time, and may offer substantial help if the student has tried several times and still can not get the correct answer).

Although these types of tutoring systems are sensitive to the students' performance and can choose different teaching strategies, they do not take any concern of the students and spend little effort to stimulate their interests of learning. Therefore, the students are very passive and the tutoring is not effective.

The second approach to conduct private tutoring encourages the student to participate in a tutoring process where the system and the student work cooperatively as a team, that is, the system sets up the teaching goals (e.g., "have the student know how to create a table") for tutoring topics in a course and the student selects the paths for going through the topics to achieve the teaching goals. During a tutoring process, the student has a certain degree of freedom to choose or eliminate topics according to his/her special learning needs, requests and interests concerning the subject materials.

SQL–TUTOR has adopted the second approach. As we have discussed in Section 2, during a planning phase, the next topic for a student to study can be either selected by the student (active mode) or generated by the system (passive mode). After a topic has been selected by the student and received by the planning module of the Communication Controller, the system will use the student model to check whether the student has mastered all the knowledge which is prerequisite to this topic. If the student's performance on all of these prerequisite topics is satisfactory, the system will generate a *learning graph*, which is a subgraph of the *TAG* for the original course, for guiding the student to study the corresponding topic. On the other hand, if the student has not mastered all the prerequisite knowledge, the system will generate a series of other learning graphs, one for each prerequisite topic, and the student has to follow these learning graphs to study the prerequisite topics before he/she could begin to learn his/her selected topic. In the following subsections, we will

2. The reader is referred to [40] for their proofs.

formulate the concept of the learning graph, discuss some of its properties, and show how to generate different learning graphs according to different learning objectives.

4.2 Incorporating Learning Goals into a Tutoring Process

A learning goal (Lgoal) is a pedagogical objective of a student for a course, which can be expressed in terms of the topics of the course. Since a tutoring process is a communication process participated by both an instructor and a student in a cooperative manner, a KBTS must take into account the student's learning objectives.

There are three kinds of Lgoals that a student can set up during a tutoring process while working with SQL-TUTOR:

1) to choose a particular topic to study (e.g., "study how to create databases and tables");
2) to study a topic from a particular organization (e.g., "study how to create databases and tables from Ullman's book"); and
3) to exclude a specific topic from his/her curriculum (e.g., "skip optimizing techniques").

SQL-TUTOR provides its students with a group of *TAG* operators (*FOCUS, SELECT, STUDY, SKIP, DELETE*) to accomplish their Lgoals. By applying these operators, the students can reconstruct their study curriculum based on individual needs. Therefore, they are relatively free to choose the best curriculum for themselves.

DEFINITION 8. *Let T_1 and T_2 be two distinguished topics in the same view and* $\mathbf{p}(T_1, T_2)$. *T_1 is said to be a* strong precedence *of T_2, denoted as* $\mathbf{sp}(T_1, T_2)$, *if for all T_i,* $\mathbf{cs}(T_i, T_1)$, *we have* $\mathbf{p}(T_i, T_2)$.

Consider again the *TAG* shown in Fig. 6. Topic `BASIC CONCEPTS` is a strong precedence of topic `SEARCH EXPRESSION`, because we have:

(1) `p(EXPRESSION, SIMPLE SEARCH EXPRESSION)`
(Given in Fig. 6)
(2) `s(SIMPLE SEARCH EXPRESSION, SEARCH EXPRESSION)`
(Given in Fig. 6)
(3) `p(EXPRESSION, SEARCH EXPRESSION)`
((1), (2), Item 3 of Definition 6)
(4) `p(CONSTANT, EXPRESSION)`
(Given in Fig. 6)
(5) `p(CONSTANT, SEARCH EXPRESSION)`
((3), (4), and the transitivity)
(6) `p(VARIABLE, EXPRESSION)`
(Given in Fig. 6)
(7) `p(VARIABLE, SEARCH EXPRESSION)`
((3), (6), and the transitivity)
(8) `sp(BASIC CONCEPTS, SEARCH EXPRESSION)`
((3), (5), (7), and Definition 8)

Likewise, we can also obtain `sp(BASIC CONCEPTS, SIMPLE SEARCH EXPRESSION)`.

DEFINITION 9. *A* learning graph *$LG = (\mathcal{N}', \mathcal{E}')$ of a $TAG = (\mathcal{N}_s, \mathcal{N}_c, \mathcal{E}_s, \mathcal{E}_p)$, where $\mathcal{N}' \subset \mathcal{N}_s \cup \mathcal{N}_c$ and $\mathcal{E}' \subset \mathcal{E}_s \cup \mathcal{E}_p$, is an induced subgraph of the TAG. It is* well defined *with respect to the TAG if for any topic $T_j \in \mathcal{N}'$ and* $\mathbf{sp}(T_i, T_j)$, *we have $T_i \in \mathcal{N}'$. A TAG operator is* well defined *if it always yields well defined learning graphs.*

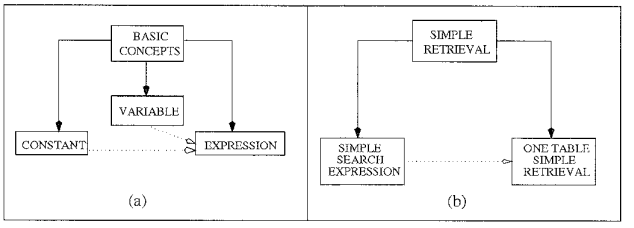


Fig. 7. Two learning graphs: (a) is well defined, but (b) is not.

Fig. 7 depicts two learning graphs generated from the *TAG* shown in Fig. 6. The first graph is well defined. However the second graph is not well defined because topic `BASIC CONCEPTS`, which is a strong precedence of topic `SIMPLE SEARCH EXPRESSION`, is not in the graph.

Intuitively, a well defined learning graph contains *all* the prerequisite contents within the course which are necessary for the study of its topics. Thus, it can be considered to be *complete* and a student can study the topics by following the curriculum defined by the learning graph. We now discuss the operations for manipulating a *TAG* in detail. Among the operators, *FOCUS, STUDY,* and *DELETE* are well defined.[3]

4.3 *FOCUS* Operator

When there is more than one view associated with a topic, a student can select a view by himself/herself or can ask the system to select a view for him/her. For the latter case, the system will make the selection based on the rules stored in its pedagogical knowledge base. For example, two of the pedagogical rules could be:

1) If $\mathcal{T}(T)$ is the view used by the student last time when he/she studied the topic T and he/she did not pass the test, then select another view this time.
2) If a view $\mathcal{T}(T)$ has been used by many students successfully, then select the view to study the topic T.

By using operator *FOCUS*, a student can select a particular view. Given a $TAG = (\mathcal{N}_s, \mathcal{N}_c, \mathcal{E}_s, \mathcal{E}_p)$ which contains a complex topic $T \in \mathcal{N}_c$ and a view $\mathcal{T}(T)$. Operation $FOCUS(TAG, \mathcal{T}(T))$ allows a student to focus only on the view $\mathcal{T}(T)$. Therefore, when there is more than one view associated with a topic, the student can select his/her preferable one.

The result of applying operation $FOCUS(TAG, \mathcal{T}(T))$ is a well defined learning graph $LG = (\mathcal{N}', \mathcal{E}')$, where $\mathcal{N}' = \{T' \mid \mathbf{s}(T', T, \mathcal{T}(T))\}$. That is, $\mathcal{N}'$ is obtained from $\mathcal{N}_s \cup \mathcal{N}_c$ by removing those topics which are not subtopics of T with respect to $\mathcal{T}(T)$. Consider again the topic `QUERYING TABLES` in the TAG shown in Fig. 6. If a student focuses on its first view labeled by `1`, then the learning graph generated is the topic tree shown in Fig. 4a.

4.4 *SELECT* Operator

Recall that a tutoring process involving a KBTS and a student can be considered as a knowledge communication between the system and the student. This kind of communication consists of a series of communication cycles, each of which focuses on one topic. The topic can be selected by

3. The reader is referred to [40] for the proofs.

the system, or by the student using operators SELECT or STUDY.

Given a $TAG = (\mathcal{N}_s, \mathcal{N}_c, \mathcal{E}_s, \mathcal{E}_p)$ and a topic T of the TAG, the operation $SELECT(TAG, T)$ generates a learning graph $LG = (\mathcal{N}', \mathcal{E}')$, where $\mathcal{N}' = \{T\} \cup \{T' \mid \mathbf{s}(T', T)\}$. The learning graph generated by a *SELECT* operation is not necessarily well defined. By using *SELECT*, a student can study a topic even though the student model indicates that he/she has not mastered all the precedence topics. If this happens, the system will tell the student that some precedence knowledge is missing from him/her and let the student decide whether he/she should go ahead or study the precedence topics first. In this way, the system works as an advisor who tells the student his/her status regarding the domain knowledge and the student selects the topic to study.

4.5 *STUDY* Operator

Given a $TAG = (\mathcal{N}_s, \mathcal{N}_c, \mathcal{E}_s, \mathcal{E}_p)$ which contains topic T, the operation $STUDY(TAG, T)$ creates a learning graph $LG = (\mathcal{N}', \mathcal{E}')$, where $\mathcal{N}' = \{T\} \cup \{T' \mid \mathbf{s}(T', T) \vee \mathbf{sp}(T', T)\}$. That is, the learning graph contains T, all the subtopics of T, and the strong precedences of T. Therefore, the student can use this learning graph as a personal curriculum to study the selected topic.

Fig. 8 shows the resulting graph obtained by applying *STUDY* to topic **ONE TABLE RETRIEVAL** in the TAG of Fig. 6. In this figure, we have divided the topics into two regions where Region I includes **ONE TABLE RETRIEVAL** and all of its subtopics and Region II includes all of the topics which are strong precedences of **ONE TABLE RETRIEVAL**.

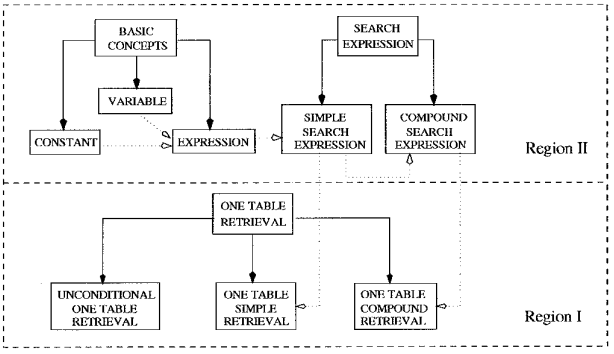


Fig. 8. The learning graph obtained by applying operator STUDY to topic **ONE TABLE RETRIEVAL** in the TAG shown in Fig. 6.

On one hand, *STUDY* is like *SELECT* which allows a student to select a particular topic to study. On the other hand, *STUDY* differs from the *SELECT* in that all of the precedence materials are also included in the resulting curriculum (i.e., the learning graph generated by the *STUDY* is always well defined).

4.6 *SKIP* Operator

Given a $TAG = (\mathcal{N}_s, \mathcal{N}_c, \mathcal{E}_s, \mathcal{E}_p)$, and a topic T of the *TAG*, the operation $SKIP(TAG, T)$ yields a learning graph $LG = (\mathcal{N}', \mathcal{E}')$, where $\mathcal{N}' = (\mathcal{N}_s \cup \mathcal{N}_c) - (\{T\} \cup \mathcal{N}_1)$ and $\mathcal{N}_1 = \{T' \mid \mathbf{s}(T', T)\}$. That is, $\mathcal{N}'$ is obtained from $\mathcal{N}_s \cup \mathcal{N}_c$ by removing T and all its subtopics. Like operator *SELECT*, the learning graph generated by *SKIP* is not necessarily well defined.

4.7 *DELETE* Operator

Given a $TAG = (\mathcal{N}_s, \mathcal{N}_c, \mathcal{E}_s, \mathcal{E}_p)$ and a topic T of the *TAG*, the operation $DELETE(TAG, T)$ allows a student to remove any optional and uninteresting topics from his/her curriculum. The operation $DELETE(TAG, T)$ generates a well defined learning graph $LG = (\mathcal{N}', \mathcal{E}')$, where $\mathcal{N}' = (\mathcal{N}_s \cup \mathcal{N}_c) - (\{T\} \cup \mathcal{N}_1 \cup \mathcal{N}_2 \cup \mathcal{N}_3)$, and

1) $\mathcal{N}_1 = \{T' \mid \mathbf{s}(T', T)\}$;
2) $\mathcal{N}_2 = \{T' \mid \mathbf{p}(T, T')\}$; and
3) $\mathcal{N}_3 = \{T' \mid \forall T'', T''$ is a unit and $\mathbf{s}(T'', T'), T'' \in (\mathcal{N}_1 \cup \mathcal{N}_2)\}$.

That is, $DELETE(TAG, T)$ removes all the topics in $\{T\} \cup \mathcal{N}_1 \cup \mathcal{N}_2 \cup \mathcal{N}_3$. Intuitively, $\mathcal{N}_1$ contains all the subtopics of T; $\mathcal{N}_2$ contains topics that take T as a precedence; and $\mathcal{N}_3$ contains topics that no longer have unit subtopics due to the removal of topics in $\{T\} \cup \mathcal{N}_1 \cup \mathcal{N}_2$.

Therefore, *DELETE* to *SKIP* is like *STUDY* to *SELECT*: it allows a student to remove some topic from his/her curriculum, and the learning graph generated is always well defined. If a student wants to remove an optional topic and everything related to the topic from his/her study curriculum, he/she can use *DELETE*. On the other hand, if the student already knows the material of a topic and wants to remove it while keeping the related materials in the study curriculum, he/she can use *SKIP*.

Fig. 9 shows the result of applying *DELETE* to the topic **SIMPLE RETRIEVAL** in the *TAG* shown in Fig. 6. Here, we have removed the following topics:

1) **SIMPLE RETRIEVAL**;
2) all the subtopics of **SIMPLE RETRIEVAL**;
3) **COMPOUND RETRIEVAL** and its subtopics, because they all take **SIMPLE RETRIEVAL** as their precedence;
4) **CONDITIONAL RETRIEVAL**, because its unit subtopics are all removed.

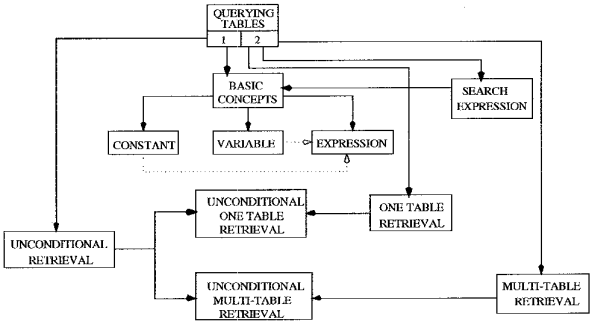


Fig. 9. The learning graph obtained by applying *DELETE* to topic **SIMPLE RETRIEVAL** in the *TAG* shown in Fig. 6.

5 Conclusions and Implementations

In this paper, we have presented the SQL-TUTOR architecture and developed a graphical notation and schema to support the manipulation of the curriculum knowledge during the planning phase of a communication cycle. We

use the Topic Association Graph (*TAG*) to represent the curriculum knowledge. There are two groups of work closely related to ours. The Domain Expert in ExperTutor [11] is constructed from a Goal/Task Hierarchy, which is a lattice of lesson components ordered by an epistemological priority relation. The higher goals in the hierarchy need more expertise than the lower goals. The student has to study and complete all subgoals before he/she can complete a goal successfully. An instructor can include several rule bases within each node of the hierarchy to provide alternative teaching styles, further examples, remediation, tests, and to determine the next action taken by the system.

Lesgold [16] found that the curriculum goal hierarchy was not powerful enough to capture the curriculum relations. He formulated a curriculum structure which has three layers of knowledge. The middle layer is the *curriculum goal lattice*, which can incorporate a number of viewpoints on the goals of the instruction. An important feature of his formulation is that the lowest level units, viz. the simple lessons, are the same from all the viewpoints. Like the Goal/Task Hierarchy, the connections between the goals are also created explicitly by the developers.

In contract to Lesgold's goal lattice, the *TAG* representation presented here allows an instructor to incorporate his/her knowledge about the goal hierarchy of a course, the multiple viewpoints on a topic, and the prerequisite relations among the topics into the curriculum knowledge base. These types of knowledge can help a tutoring system select appropriate topics to teach and diagnose the student's mistakes. A novel feature of our formulation is that the precedence relations among topics are derived from the prerequisite relations over the domain concepts. The system is capable of checking their consistency during the derivation process. Thus, our approach provides a framework for an instructor to explicitly encode his/her curriculum knowledge in the form of a *TAG*.

We have implemented the SQL-TUTOR in C and Tcl-Tk [25] on a Sun SPARCstation 20 run under the operating system Solaris version 2.4. Through the interface of the system, the instructor can construct the various knowledge bases used by our system [8], [28] and the student can learn the SQL programming [15], [40]. Work on SQL-TUTOR is continuing. Our main goals include

- the development of intelligent tutoring systems for other domains (e.g., Prolog programming) and examining the feasibility of the proposed framework for these other domains;
- the development of a class of qualitative and quantitative measures for evaluating the usability and effectiveness of our tutoring system and conducting experiments to compare our approach with other methods, such as classical and collaborative teaching, as well as distance learning systems [14].

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their constructive criticism and insightful comments, which not only helped us improve the quality and presentation of the paper, but also gave us ideas for future research. This work was supported in part by the National Science Foundation under grant IRI-9224602, by the New Jersey Institute of Technology under grant SBR-421280, and by a grant from the AT&T Foundation.

REFERENCES

[1] J.R. Anderson, C.F. Boyle, and G. Yost, "The Geometry Tutor," *Proc. Ninth Int'l Joint Conf. Artificial Intelligence*, pp. 1–7, Los Angeles, 1985.

[2] B.S. Bloom, "The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring," *Educational Researcher*, vol. 13, pp. 3–16, 1984.

[3] E. Bos, "Error Diagnosis in a Tutoring System for the Conjugation and Spelling of Dutch Verbs," *Computer in Human Behavior*, vol. 10, pp. 33–49, 1994.

[4] B.J. Brecht, G.I. McCalla, J.E. Greer, and M. Jones, "Planning the Content of Instruction," *Artificial Intelligence and Education: Proc. Fourth Int'l Conf. AI and Education*, D. Bierman, J. Breuker, and J. Sandberg, eds., pp. 32–41. Amsterdam. Springfield, Va: IOS, 1989.

[5] M.L. Burger and J.F. Desoi, "The Cognitive Apprenticeship Analogue: A Strategy for Using ITS Technology for the Delivery of Instruction as a Research Tool for the Study of Teaching and Learning," *Int'l J. Man-Machine Studies*, vol. 36, pp. 775–795, 1992.

[6] A. Cawsey, "The Structure of Tutorial Discourse," *Artificial Intelligence and Education, Proc. Fourth Int'l Conf. AI and Education*, D. Bierman, J. Breuker, and J. Sandberg, eds., pp. 47–53. Amsterdam. Springfield, Va.: IOS, 1989.

[7] T.W. Chan, "Curriculum Tree: A Knowledge-Based Architecture for Intelligent Tutoring Systems," *Intelligent Tutoring Systems: Second Int'l Conf. Intelligent Tutoring Systems, ITS '92*, C. Frasson, G. Gauthier, and G.I. McCalla, eds., pp. 140–147. Springer-Verlag, 1992.

[8] S. Chang, "Algorithms for Manipulating Topic Association Graph in Knowledge-Based Tutoring Systems," master's project, Dept. of Computer and Information Science, New Jersey Inst. of Technology, Dec. 1994.

[9] W.J. Clancey, "Tutoring Rules for Guiding a Case Method Dialogue," *Int'l J. Man-Machine Studies*, vol. 11, pp. 25–49, 1979.

[10] W.J. Clancey, *Knowledge-Based Tutoring: The GUIDON Program*. Cambridge, Mass.: MIT Press, 1983.

[11] N.G. Craske, T. Richards, and G. Cumming, "ExperTutor, a Generic Purpose Intelligent Educational System," *Advanced Research on Computer in Education*, R. Lewis and S. Otsuki, eds. North-Holland: Elsevier Science Publishers B. V., 1991.

[12] C. Dede, "A Review and Synthesis of Recent Research in Intelligent Computer-Assistant Instruction," *Int'l J. Man-Machine Studies*, vol. 24, pp. 239–353, 1986.

[13] J. Dessalles, "Computer Assisted Conceptual Learning," *Computer Assisted Learning, Third Int'l Conf., ICCAL '90*, G. Goos and J. Hartmanis, eds., pp. 175–183, Hagen, France, 1990.

[14] S.R. Hiltz, *The Virtual Classroom: Learning without Limits via Computer Network*. Norwood, N.J.: Ablex Publishing, 1994.

[15] S. Jagarlamudi, "Interface for the Curriculum Knowledge Base Management in SQL–TUTOR," master's project, Dept. Computer and Information Science, New Jersey Inst. of Technology, Dec. 1994.

[16] A. Lesgold, "Toward a Theory of Curriculum for Use in Designing Intelligent Instructional Systems," *Learning Issues for Intelligent Tutoring Systems*, H. Mandl and A.M. Lesgold, eds., pp. 114–137. New York: Springer-Verlag, 1988.

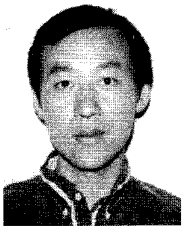
[17] P. Marcenac, "An Authoring System for ITS Which is Based on a Generic Level of Tutoring Strategies," *Computer Assisted Learning, Fourth Int'l Conf., ICCAL '92*, T. Tomek, ed., pp. 428–440, Nova Scotia, Canada, 1992.

[18] K.V. Marcke, "Instructional Expertise," *Intelligent Tutoring Systems: Second Int'l Conf. Intelligent Tutoring Systems, ITS '92*, C. Frasson, G. Gauthier, and G.I. McCalla, eds., pp. 234–243, 1992.

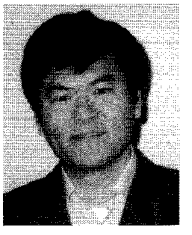
[19] J. Mcdonald, "The EXCHANGE CAI System," *University-Level Computer-Assistance Instruction at Stanford: 1968-1980*, P. Suppes, ed.. Inst. for Mathematical Studies in the Social Sciences, Stanford Univ., Stanford, Calif., 1981.

[20] R. Moyse, "Knowledge Communication Implies Multiple Viewpoints," *Artificial Intelligence and Education: Proc. Fourth Int'l Conf. AI and Education*, D. Bierman, J. Breuker, and J. Sandberg, eds., pp. 140–149, Amsterdam, May 1989. Springfield, Va.: IOS.

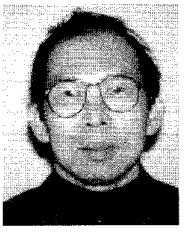
[21] R. Moyse and M. Elson-Cook, "Knowledge Negotiation: An Introduction," *Knowledge Negotiation*, R. Moyse and M. Elson-Cook, eds., pp. 1–19. Academic Press, 1992.
[22] W.R. Murray, "Control for Intelligent Tutoring Systems: A Blackboard-Based Dynamic Instructional Planner," *Artificial Intelligence and Education: Proc. Fourth Int'l Conf. AI and Education*, D. Bierman, J. Breuker, and J. Sandberg, eds., pp. 150–168. Amsterdam, May 1989. Springfield, Va.: IOS.
[23] W.R. Murray, "A Blackboard-Based Dynamic Instructional Planner," *Proc. Eighth Nat'l Conf. Artificial Intelligence*, pp. 434–441, Boston, 1990.
[24] H.S. Nwana, "FITS: A Fraction Intelligent Tutoring System," *Proc. 13th Nat'l Conf. Artificial Intelligence*, pp. 49–54, 1991.
[25] J.K. Ousterhout, *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.
[26] D.R. Peachey and G.I. McCalla, "Using Planning Techniques in Intelligent Tutoring Systems," *Int'l J. Man–Machine Studies*, vol. 24, pp. 77–98, 1986.
[27] B.J. Reiser, J.R. Anderson, and R.G. Farrell, "Dynamic Student Modeling in an Intelligent Tutor for LISP Programming," *Proc. Eighth Int'l Joint Conf. Artificial Intelligence*, pp. 8–14, Los Angeles, 1985.
[28] C. Sarda, "Implementation of Domain Knowledge Base Management for a Knowledge-Based Tutoring System," master's project, Dept. of Computer and Information Science, New Jersey Inst. of Technology, Dec. 1994.
[29] C.B. Schwind, "An Intelligent Language Tutoring System," *Int'l J. Man-Machine Studies*, vol. 33, pp. 557–579, 1990.
[30] V.J. Shute, "Regarding the I in ITS: Student Modeling," *Proc. ED-MEDIA 94—World Conf. Educational Multimedia and Hypermedia*, pp. 80–87, Vancouver, BC, Canada, 1994.
[31] D.H Sleeman and J.S. Brown, *Intelligent Tutoring Systems*. London: Academic Press, 1982.
[32] J.C. Spohere and E. Soloway, "Simulating Student Programmers," *Proc. Ninth Int'l Joint Conf. Artificial Intelligence*, pp. 543–549, Detroit, 1989.
[33] N.A. Streitz, "Mental Models and Metaphors: Implications for the Design of Adaptive User-System Interface," *Learning Issues for Intelligent Tutoring Systems*, L. Mandl, ed., pp. 164–186. New York: Springer-Verlag, 1988.
[34] T. Tokuda and A. Fukuda, "A Probabilistic Inference Scheme for Hierarchical Buggy Models," *Int'l J. Man-Machine Studies*, vol. 38, pp. 857–872, 1993.
[35] E. Wenger, *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*. Los Altos, Calif.: Morgan Kaufmann, 1987.
[36] B.Y. White and J.R. Frederiksen, "Modeling Expertise in Trouble-Shooting and Reasoning about Simple Electric Circuit," *Proc. Sixth Cognitive Science Conf.*, pp. 337–343, Boulder, Colo., 1984.
[37] B.Y. White and J.R. Frederiksen, "Intelligent Tutoring Systems Based upon Qualitative Model Evaluations," *Proc. Eighth Nat'l Conf. Artificial Intelligence*, pp. 313–319, Philadelphia, 1986.
[38] B. Woolf, D. Blegen, J. Jansen, and A. Verloop, "Teaching a Complex Industrial Process," *Proc. Eighth Nat'l Conf. Artificial Intelligence*, pp. 722–728, Philadelphia, 1986.
[39] Y. Yano, A. Kashhara, and W. McMichael, "Stabilizing Student Knowledge in Open Structured CAI," *Int'l J. Man-Machine Studies*, vol. 37, pp. 595–612, 1992.
[40] G. Zhou, "Towards Designing a Knowledge-Based Tutoring System: SQL-TUTOR as an Example," PhD thesis, Computer and Information Science Dept., New Jersey Inst. of Technology, 1996.
[41] G. Zhou, J.T.L. Wang, and P.A. Ng, "A Knowledge-Based Tutoring System for SQL Programming," *Proc. Sixth IEEE Int'l Conf. Tools with Artificial Intelligence*, pp. 352–358, New Orleans, 1994.



Gang Zhou received the BS and MS degrees in 1981 and 1984, respectively, in computer science from Jilin University, People's Republic of China. He is currently a PhD candidate in the Computer and Information Science Department at the New Jersey Institute of Technology. His research interests include computer tutoring systems, knowledge representation and reasoning, logic programming, deductive databases, and operating system design theory.



Jason Tsong-Li Wang received the BS degree in mathematics from National Taiwan University, and the PhD degree in computer science from the Courant Institute of Mathematical Sciences at New York University in 1991. He is currently an associate professor in the Computer and Information Science Department at the New Jersey Institute of Technology and director of the university's Data and Knowledge Engineering Laboratory. He is also a member of the PhD in Management faculty at the Graduate School of Rutgers University, Newark, New Jersey. Dr. Wang's research interests include data and knowledge management systems, multimedia information retrieval, software development, pattern discovery, and computational biology. He has authored or coauthored 50 publications in these fields. He serves on the Editorial Advisory Board of *Information Systems*, is listed in *Who's Who Among Asian Americans* (1994), and is a member of ACM, IEEE, AAAI, and SIAM.



Peter A. Ng received the PhD degree in computer science from the University of Texas at Austin in 1974. He is a professor and chairman of the Department of Computer and Information Science at the New Jersey Institute of Technology, Newark, New Jersey. He is also director of the Institute for Integrated Systems Research. He is currently involved in the development of the Text Processing System (TEXPROS). With Dr. Raymond Yeh, he is a cofounder of the International Conference on Systems Integration (ICSI) in 1990. He is the founder and an editor-in-chief of the *Journal on Systems Integration*, and an advisory editor for *Data and Knowledge Engineering*, North Holland. His research interests include information and document processing systems, software engineering, database management systems, distributed processing systems, and computer visions. He is a member of the IEEE Computer Society and the Association for Computing Machinery.