Compilation of Domain Principles into Inference Operators for Science Tutoring

Hyung Joon Kook

Department of Computer Engineering, Sejong University, Seoul, Korea
kook@sejong.ac.kr

Abstract

The nature of learning in science is to study the principles of the domain and apply them to solving problems. Consequently, a tutoring system in a scientific domain should possess an adequate methodology to deal with this principle. Due to the diversified contexts of various scientific domains, however, the techniques developed for tutoring in one scientific domain do not necessarily transfer well to other domains. We developed a tutoring architecture for geometry where the domain principles are automatically converted to *inference operators* for use by the domain-independent, inferential portion of the tutoring system, which automatically generates the solutions to problems at tutoring time based on the inference operators. The system also has a potential for providing personalized feedback. The proposed design is expected to apply to additional tutoring domains other than geometry, where problem solving is also based on stepwise application of domain principles.

Keywords: Science tutoring, inference operators, knowledge compilation.

1 Introduction

Learning in science is a process made up of studying a finite set of general principles and then developing the skills to apply them in an infinite variety of situations. A computer tutor in science should, therefore, possess an adequate strategy to represent and to use the principles in tutoring. Although such a strategy is expected to be commonly applicable to various scientific domains, a technique developed for tutoring in a domain cannot always be applied directly to other domains because each scientific domain is taught and learned in a context radically different from any other. For example, learning in mechanics involves reasoning about physical objects, motions, forces, etc., while the main contexts of plane geometry are figures, angles, etc.

We have set as our long-term research goal the development of a modular science tutoring architecture, in which the common, sharable strategies of science tutoring are built in and modifiable independently of the domain-specific knowledge supplied by the domain authors. We believe that authoring of domain-specific knowledge must be

facilitated in a human-friendly interface that is designed to minimize the burden on the human author who supplies the knowledge (e.g., concepts and principles) of the domain. More importantly, the requirement for the tutoring architecture suggests that the system should provide a mechanism for compiling the knowledge of the domain into a format suitable for tutoring during the reasoning part of the teaching process.

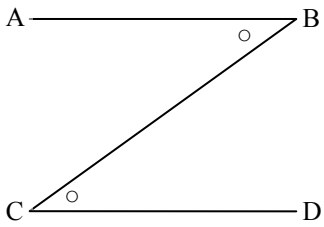
As an important step toward our long-term research goal, we have designed one such compilation mechanism and built on it with a tutoring system for geometry problem solving called CyberTutor. In our design, stored principles in geometry are combined with a specific problem (selected for tutoring) to automatically produce a homogeneous set of logical consequences, called *inference operators*. These operators are subsequently used by the domain-independent, inferential component of the system for tutoring on that specific problem. This paper reports on the design and the implementation of the tutoring system with a focus on the working mechanisms of the inference operators and the benefits of using them.

The essence of the proposed tutoring architecture is also expected to be applicable to other scientific domains. Although geometry has been a popular domain for numerous intelligent tutoring systems research [1], [6], [7], few of the system have yet reached the level of building a modular architecture like the one we are developing. A major obstacle to such modularization seems to be that, in geometry the domain knowledge is taught and learned in a unique context, i.e., *the figure*, an application that is uncommon in other domains. In the following sections, we will show how the geometry principles exemplified by figures can be represented in the computational framework, and how the principles can be compiled into a set of inference operators to use in the inferential part of the tutoring system. Next, we will present the working mechanism of the inferential components of the tutoring system. That aspect will be followed by a demonstration of a sample tutoring session by CyberTutor. Finally, after a discussion of some future research, conclusions will be presented.

2 Modeling Geometry Principles

As in other scientific domains, principles are the main contents of learning in geometry. Students are taught to study them and then apply them to solving various problems. Unlike other domains, geometry principles are usually presented with figures to graphically describe the contexts of the principles. Therefore, it is crucial for any computer geometry tutor to be able to deal properly with figures.

A geometry principle is usually stated as a logical proposition about a figure representing the context in which the principle is applicable. Such a figure contains a set of geometrically-meaningful information, which may be divided into its components, quantities and relations. The components are the objects, such as points, lines, and figures like triangles and circles. The quantities are the quantitative attributes associated with the components, such as lengths, angles, perimeters, and areas. The relations



"Alternate interior angles between parallel lines are equal"

Fig. 1a. A Geometry Principle

```
(MODEL M25
    (PRINCIPLE "Alternate Interior Angles between Parallel Lines
                are Equal")
    (COMPONENTS
        (POINTS (A) (B) (C) (D))
        (LINES (AB (ENDS A B)) (BC (ENDS B C)) (CD (ENDS C D))))
    (QUANTITIES
        (ANGLES (ABC (SIDES AB BC)) (BCD (SIDES BC CD))))
    (RELATIONS
        (SHIFT (DIRECTION AB BC))
        (SHIFT (DIRECTION CD BC)))
    (PROPOSITION
        (AB // CD → ABC = BCD))
)
```

Fig. 1b. A Model Corresponding to the Principle in Figure 1a

are the configurations among the components, such as 'Point A is on Circle C' and 'Direction of Line AB is shifted to the right from that of line BC'. The collection of the components, the attributes and the relations may be referred to as *contexts*, since they serve as the contextual backgrounds for the higher-level concepts and propositions of the domain.

A natural way to represent a geometry principle in a computational framework is to employ the notion of *model*, which is a declarative data structure with slots to hold the contexts as well as the proposition. These models correspond to the notion of the models in the ACT-R theory of [1]. Figure 1a shows an example principle, followed by the corresponding model in Figure 1b. It is important that an authoring system facilitate an automated environment for generating models, since manually encoding a large number of these models would be too tedious and time-consuming. The drawing

interface we are building enables an author to supply geometry principles by drawing figures and entering propositions in a user-friendly way; i.e., the contexts of the models are specified automatically, as the author draws the figures using a series of mouse actions [4].

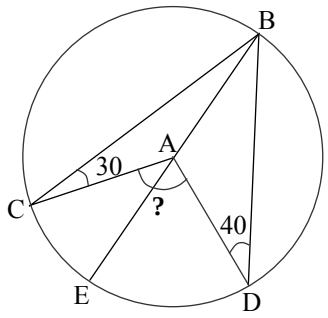
3 Compilation of Models into Inference Operators

As an embodiment of a general principle, a model should be applicable to many problems that can be presented in terms of a various set of labels and constant values. For the finite set of general models to deal effectively with an infinite variety of problems, the notion of *knowledge compilation* [2] is employed. Here the models are *compiled upon* the specific contexts of a problem to produce a set of *inference operators*. These operators are context-specific since they are obtained by converting the general contexts used in the model into the particular contexts of a problem (e.g., labels used in the figure). The compilation process applies only to those models whose contexts match the problem contexts. As a consequence, the range of the principles covered by the resultant operators is confined to only those principles that are *contextually* relevant to that specific problem.

The contexts having been matched already, these operators need not carry contextual information and thus can consist only of the propositions in the forms instantiated for the problem at hand. Figure 2b shows some of the inference operators produced for the sample problem in Figure 2a. We can see that the operators shown here solely consist of compilations from relevant models, corresponding to such principles as *"(P1) Alternate interior angles between parallel lines are equal", "(P2) If two sides of a triangle are equal, then the base angles are equal", "(P3) The sum of the angles in a triangle is 180", "(P4) Lengths of two radii of a circle are equal", "(P5) The sum of two supplementary angles is 180", "(P6) Exterior angle of a triangle is equal to the sum of opposite inner angles"* etc. Note that, some operators included in this operator set are apparently unusable ones (e.g., OPR1,2,4,5,7,8), since their left-hand side propositions do not hold. Still these operators are legitimate as far as the contexts are concerned and in fact are useful for capturing misconceptions in the inference steps that students can take.

4 The Inference Engine

The inference operators that are obtained resemble the production rules in [1], as both are derived from declarative knowledge of the domain. But the operators are tactically more useful in several respects. First of all, the search space is greatly reduced since



Given ∠ACB = 30 and ∠ADB = 40, find ∠CAD.

Fig. 2a. A Sample Problem

(OPR1)	*BC // AE → ∠ACB = ∠CAE*	*(P1)*
(OPR2)	*BD // AE → ∠ADB = ∠DAE*	*(P1)*
(OPR3)	*AB = AC → ∠ABC = ∠ACB*	*(P2)*
(OPR4)	*AC = BC → ∠BAC = ∠ABC*	*(P2)*
(OPR5)	*BC = AB → ∠ACB = ∠BAC*	*(P2)*
(OPR6)	*AB = AD → ∠ABD = ∠ADB*	*(P2)*
(OPR7)	*BD = AB → ∠ADB = ∠BAD*	*(P2)*
(OPR8)	*AD = BD → ∠BAD = ∠ABD*	*(P2)*
(OPR9)	*∠ABC + ∠ACB + ∠BAC = 180*	*(P3)*
(OPR10)	*∠ABD + ∠ADB + ∠BAD = 180*	*(P3)*
(OPR11)	*AB = AC*	*(P4)*
(OPR12)	*AB = AD*	*(P4)*
(OPR13)	*AC = AD*	*(P4)*
(OPR14)	*∠BAC + ∠CAE = 180*	*(P5)*
(OPR15)	*∠BAD + ∠DAE = 180*	*(P5)*
(OPR16)	*∠CAE = ∠ACB + ∠ABC*	*(P6)*
(OPR17)	*∠DAE = ∠ADB + ∠ABD*	*(P6)*

Fig. 2b. Some of the Inference Operators Produced for the Problem in Figure 2a

the models (i.e., the principles) that are irrelevant to the sample problem have been pruned out during compilation.

In fact, a typical problem in a scientific domain, including geometry, can usually be solved using only a few principles. Additionally, the time-taking process of context matching is completed with compilation; hence the interactive performance of the

system is enhanced still further. In this section we will describe how these operators are used in the inference processes.

CyberTutor employs three types of inference strategies: *propositional*, *algebraic*, and *quantitative*. The *propositional inference* is the common backward inference about the propositions in the working memory: i.e., the inference operators and the set of propositions given in the problem. When a student enters a problem-solving step in the form of an assertion, that student's step is taken as a hypothesis and set as the goal to be proved by the system. In the goal-driven search, the propositions in the working memory are then searched for matches. For those inference operators whose RHS's match the goal, the system tries to match their LHS propositions, possibly by sub-goaling them until the initial goal is proved or unproved. If proved, the student's step is taken as valid; otherwise, it is considered to be invalid.

Besides propositions, real problem solving in geometry often involves algebra. Students may enter a step in the form of an algebraic expression which is not derivable from a direct application of the propositional inference described above. Such a situation is handled by the *algebraic inference* mechanism of CyberTutor. For instance, a student's step, "X = 30" is taken as valid if the propositions in the working memory include, e.g., "X + Y = 180" and "Y = 150".

The algebraic inference takes more time than the propositional inference does, since it involves solving simultaneous equations. In order to minimize performance degrade, this inference is used only as a complement to the propositional inference. In particular, it is employed only when the following two conditions are met: first, that the propositional inference cannot proceed any further, and secondly, that there has been a change in the working memory since the last algebraic inference was performed (i.e., if there is nothing new in the working memory, performing the algebraic inference would also bring about nothing new).

If both the propositional and the algebraic inferences fail to validate the goal hypothesis, the system resorts finally to the *quantitative inference*, provided that the goal hypothesis consists of an algebraic expression involving quantities. In such a case, the system extracts the quantities from the expression and sets each quantity as the sub-goal of the subsequent inference. For the sub-goal quantity generated, the inference proceeds as follows. If the sub-goal quantity is X and there is a proposition in P(X) form, i.e., consisting solely of X (e.g., "X = 30") in the working memory, then the sub-goal is satisfied (i.e., solved for).

If no proposition exists in P(X) form, two alternative methods are attempted. The first method applies when there is a proposition in P(X,⋯) form, i.e., consisting of X and other quantities (e.g., "Z = X – Y"). In this case, the inference proceeds using those other quantities (e.g., Y and Z) as sub-goals. In other words, a detour is taken when the inference on the desired quantity comes to a dead end. The second method applies when there is an inference operator whose RHS proposition consists of the quantity X. In this case, that same RHS proposition of the operator is set as the sub-goal hypothesis. This strategy is useful since the chances are that validating this sub-

goal hypothesis would probably help to solve for the desired quantity. For example, when the desired quantity is X and the inference operator is "LHS → X = Y", setting "X = Y" as the sub-goal hypothesis and subsequently validating it would probably be a one-step progression to solve for X.

Considering that geometry problems often accompany algebra, it is indispensable for the system to handle unknown quantities in the methods described above. As we cannot determine which of the two methods are better, both methods are employed so as to work independently of each other in the present implementation.

Previous research on geometry tutoring has concentrated mainly on purely-proving types of problems, while ignoring the algebraic aspects involved in problem solving. In fact, many geometry problems often demand algebraic manipulations in the course of the problem solving, and some problems even ask to solve for the value of an unknown quantity. Using a separately built-in algebraic manipulation module would not suffice, since such a "black box" module would not allow the tutor to access the information about the algebraic inference performed within it. One of the achievements of our research is the generalization of the tutoring scope to cover problems involving the algebraic and the quantitative inferences in a way that is transparent, and therefore, accessible for tutoring.

5 Demonstration of CyberTutor

Based on the diverse inferential strategies presented in the preceding section, CyberTutor provides the student with an interactive environment for geometry problem solving. The interface is shown in Figure 3. As the student chooses a problem, it loads into the system. At load-time, the problem is compiled with the system's library of geometry models to produce a set of inference operators. Then, the problem tutoring session begins. It is worth mentioning here that we are in the process of developing a graphic interface that will facilitate the student entering a problem outside the library of problems, i.e., one supplied by the student himself. The strength of the proposed tutoring framework is that it is applicable not only to the selection of stored problem, but also to a wider range of problems, since the solution procedures for the problems are not stored along with the problems in the library, but generated dynamically during the problem solving.

As shown in Figure 3, a problem tutoring session comprises a series of stepwise interactions between the student and the tutor. Student problem solving is monitored so that each step entered by the student is checked for validity. This validity check is performed by using the backward inference mechanism on the propositions in the working memory, and if necessary, the algebraic and quantitative inference mechanisms as well.

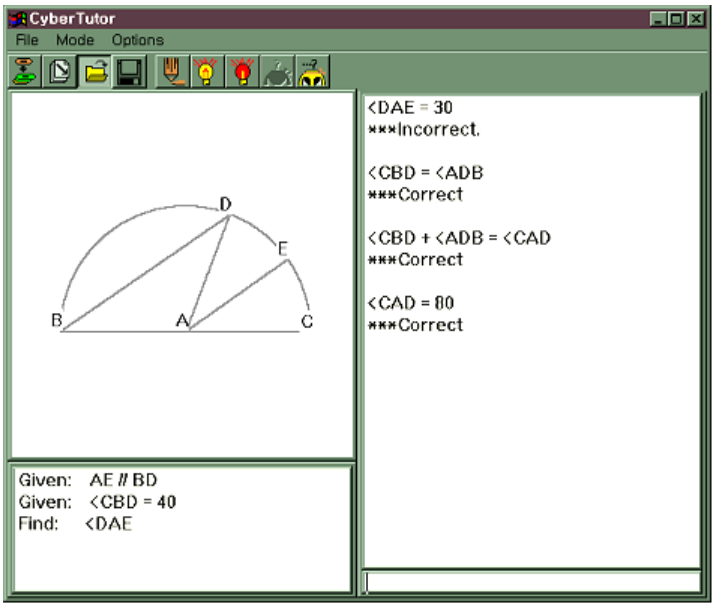


Fig. 3. CyberTutor Interface

Benefits of using the proposed design framework include the system's ability to adapt to the differing skill levels of students. A less-skilled student's step is usually traced by a single step or two inference steps of the system. On the other hand, a step involving a longer chain of inferences made by a more skilled student can also be traced by applying multiple inference steps. Most of the previous systems in this area of skills tutoring have confined the steps to a primitive length of inference, and by doing so, they failed to model the variety of individual skill levels in this respect.

6 Future Research

At the present time, the design and implementation of our tutoring system concentrates on building basic blocks: the knowledge compilation schemes, the inferential frameworks, and the skeletal tutoring architecture built around core design principles. There are other research areas that we are working on currently to improve and extend its functionality.

In an interactive tutoring environment, it is important for the system to be able to track student steps and then provide appropriate feedback [3], [5]. In its present implementation, CyberTutor checks each student step for its validity, and the result of that check is simply reported to the student. We are in the process of improving the system to offer a help when requested by the student. A tutor's step may serve as a

good help as long as it is given at the right level of the student's skill ability. In doing so, justification of a tutor's steps should also be facilitated.

We believe the proposed system design is well suited to accommodate such feedback facilities. Using its ability to solve the problem, the system can be made to generate partial steps of problem solving, hopefully at a level corresponding to the student's skill level. When the student asks for an explanation of the tutor's step, the system can respond with the operator used for generating the step. However, the explanation mechanism may have to be repeated, if requested, by back-chaining on the operators used. Below is one possible dialog segment, in which the tutor responds to the student's *why* question with an explanation derived from (OPR1) in Figure 2b.

Student: Hint please.
Tutor: Angles ∠ADB and ∠DAE are equal.
Student: Why?
Tutor: Alternate interior angles between parallel are equal.
It is given that lines AE and BD are parallel.
Therefore, angles ∠ADB and ∠DAE are equal.

Another area of improvement is the escape from the library of stored problems. Students sometimes want to work with new problems from outside the library. Authors of the problems don't want to provide various solution procedures along with the problems they supply, because doing so is tedious and error-prone. Accommodation of both demands depends on the tutoring system's capability for automatic problem-solving and the interfacial supports. Within the proposed tutoring framework, automatic problem solving is naturally supported by the inferential part of the system. A user-friendly, graphical interface is under development with the aim of supporting students and authors in entering such problems.

7 Concluding Remarks

We have reported a preliminary, but significant, result regarding building a tutoring architecture for geometry tutoring. To tutor a problem, the problem is first compiled with the domain principles stored in models to produce a set of inference operators. Then, these operators are passed to the inference part of the system, which consists of a set of domain-independent problem-solving strategies, namely propositional, algebraic, and quantitative inference strategies. They work together to assist the system's tutoring activities in a way accessible for pedagogy.

A typical solution to a science problem consists of only a few concepts and principles of the domain. A stored solution approach would be both unwise and useless. It is unwise because the human author has the burden of specifying all problem-solving steps for each problem, and remains useless because the system is still not prepared to

deal adequately with the off-the-track steps students can take. In the proposed design, the solution steps are not pre-stored in the library, but rather generated automatically at tutoring time in the form of the inference operators.

The system's capability to track student steps, including a step made of multiple inferences, is important for building individual models of students. In this respect, we believe the proposed design has a solid potential for providing a personalized learning environment, if combined with an appropriate feedback facility. Overall, the proposed design principle is expected to be applicable to various scientific domains other than just geometry, where the nature of problem solving is a stepwise inference that also uses domain principles as the search operators.

References

[1] Anderson, J. R., Boyle, F., Corbett, A., Lewis, M.: Cognitive Modeling and Intelligent Tutoring. Artificial Intelligence Vol. 42 (1990), pp. 7-49.

[2] Cadoli, M., Donini, F. M.: A Survey on Knowledge Compilation. AI Communications 10: 3-4 (1997), pp. 137-150.

[3] Koedinger, K. R., Anderson, J. R.: Reifying Implicit Planning in Geometry: Guidelines for Model-Based Intelligent Tutoring System Design. In Lajoie, S. P., Derry, S. J. (eds.): Computers as Cognitive Tools. Erlbaum (1993), pp. 15-45.

[4] Kook, H. J.: Automatic Specification of Figures in Geometry for Tutoring. In MICAI-2005: The 4th Mexican International Conference on Artificial Intelligence (2005) (Submitted).

[5] Matsuda, N., VanLehn, K.: Modeling Hinting Strategies for Geometry Theorem Proving. In UM-2003: Proceedings of the 9th Conference on User Modeling (2003), pp. 373-377.

[6] McDougal, T., Hammond, K.: Representing and Using Procedural Knowledge to Build Geometry Proofs. In AAAI-1993: Proceedings of the 11th National Conference on Artificial Intelligence (1993), pp. 60-65.

[7] Mitrovic, A., Koedinger, K. R., Martin, B.: A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modeling. In UM 2003: Proceedings of the 9th Conference on User Modeling (2003), pp. 313-322.

Hyung Joon Kook received B.S. degree from Seoul National University in 1979, and received M.S. and Ph.D. degrees in Computer Science at the University of South Carolina at Columbia and the University of Texas at Austin, USA, in 1983 and 1989, respectively. Since 1989, he has been with the Department of Computer Engineering, Sejong University in Seoul, Korea, where he currently is an associate professor. His research interests are in the areas of artificial intelligence, expert and knowledge-based systems, intelligent agents and intelligent tutoring systems.