A Reification of a Strategy for Geometry Theorem Proving

Noboru Matsuda
Intelligent Systems Program
University of Pittsburgh
mazda@isp.pitt.edu

Kurt VanLehn*
Learning Research and Development Center
University of Pittsburgh
vanlehn@cs.pitt.edu

3939 O'Hara Street, Pittsburgh, PA 15260 USA

Abstract

This paper addresses the design issues of learning environment for problem solving in a complex domain, such as geometry theorem proving with construction. We first discuss the difficulties to study problem-solving strategies (i.e., forward and backward inference, and backing-up at impasse) in such a domain, and examine the three kinds of representations to describe a problem solving with search; viz., a proof tree, a search tree, and a problem tree. We then claim that it is a search tree that we need to reify to teach students the problem-solving strategies. Finally, we show a geometry learning environment with a GUI that reifies a search to prove geometry theorems with auxiliary line constructions. Several powerful scaffolding techniques in this learning environment are discussed followed by a conclusion that our reification technology might have a potential ability to facilitate students to learn desired problem-solving strategies.

1 Introduction

When one wishes to teach students in a complex problem-solving domain, it is desired to provide students with better models of how to solve problems. Those models should be visualized and manipulatable; that is, we need a better reification of the domain. Especially, a reification of problem-solving strategies is one of the key issues that affect the quality of instructions in such complex domains. By "problem-solving strategies" we mean domain independent knowledge to conduct searches through a problem space. It might include carrying out a backward or a forward inference step, backing-up an alternative inference at a dead-end, switching over backward and forward inference, and so forth.

Despite the complexity of the domain, most of the time, when we design an intelligent tutoring system with an articulate expert system, we believe that since the built-in expert

*This research was supported by NSF grant number 9720359 to CIRCLE: Center for Interdisciplinary Research on Constructive Learning Environments. http://www.pitt.edu/~circle

system holds a "cognitive model," observing and/or working along the embedded cognitive model is enough for students to understand the domain. This design policy appears as a form of hints generated by the ITS or a form of user interface. That is, it might be assumed, for example, that showing a problem-solving tree is good enough to explain how to solve a problem. It might also be assumed that providing a list of possible operators are good enough for students to learn the domain by solving problems. However, careful observation of the inside of the problem-solving strategies implies the difficulties for students to learn those strategies as well as the design issues to build ITSs.

In this paper, we focus on the procedures to carry out backward and forward inferences. From an AI standpoint, they just tell how to apply operators — for a forward inference, seek an operator that has all the preconditions satisfied and execute the action part, and for a backward inference, seek an operator that has an effect matched against a current goal and establish subgoals with the unsatisfied preconditions. From a cognitive and instructional standpoint, however, they are rather much complicated as discussed below.

First, carrying out a backward inference might require nondeterministic decision making and complex cognitive procedures. Given a set of goals to be achieved in the goal stack ($\mathbf{G}$) and a set of operators ($\mathbf{OP}$), a backward inference consists of following basic steps.

1. Select a single goal $\boldsymbol{g} \in \mathbf{G}$, and remove $\boldsymbol{g}$ from $\mathbf{G}$.
2. Match $\boldsymbol{g}$ against the known facts in the current state. If such fact is found, then the goal is achieved.
3. If the previous step fails, then *choose* an operator $\boldsymbol{op} \in \mathbf{OP}$ whose effect contains $\boldsymbol{g}$ and put the preconditions of $\boldsymbol{op}$ onto $\mathbf{G}$.
4. If no $\boldsymbol{op}$ is available at the previous step, then back-up; viz., *choose* another operator for an ancestor of the current goal.

At the third step, one needs to *choose* an operator, but most of the time it is not deterministic (indeed, if it is always deterministic, nothing is fun to solve problems!). So, one must estimate the fitness of the operator and envision the success of its application. However, such estimation and envisioning are not easy tasks for human being. We need a scaffolding technique for them.

At the fourth step, one needs to overcome an impasse by seeking an alternative operator. This procedure requires one to determine where to back-up. However, since there might be several points to be able to back-up in a complex problem, to deterine where to back-up requires to maintain a memory in a systematic way which, apparently, is not easy for novices. Thus, it is desired to reify a backing-up procedure.

Second, showing a problem-solving tree might not help students to understand a process of problem solving. To explain why, we need to break down the "tree" into meaningful pieces;

[[SIMPLE EXAMPLES ARE SHOWN HERE]]

Figure 1: An example of the trees.

- A **proof tree** is an AND-tree to show a proof. [1]
- A **search tree** is an OR-tree to show how to search a solution.
- A **problem tree** is an AND/OR-tree to show an entire search space.

Figure 1 shows an example of those trees on a tiny problem with four operators. Most ITSs utilize a problem tree. For example, Algebra Tutor [**?**], Geometry Tutor [2], Related Rates Tutor (a calculus tutor) [9], Angle (a geometry tutor) [3], and GIL (a LISP tutor) [5] all offer students a partial paroblem tree to build up a solution. **** NEED FOR A CITATION FOR THE ALGEBRA TUTOR ****

The major problems of using the problem tree as a direct media of instruction include, among other things, (a) since a problem tree is mixing up AND-trees and OR-trees which in turn results in a big tree, it is virtually confounding students to see what to see, and thus an attention tends to be loose; in other words, it is difficult to read necessary information off the tree, (b) since chronological information is not embedded into the tree, it is difficult to see how the search is conducted, and (c) back-up is not shown at all.

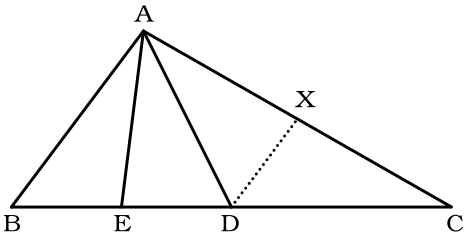
Although the goal scaffolding, a version of reification of a proof tree, facilitates students to learn how to conduct backward inference, it does not help students learn backing-up procedures at all. **** NEED CITATION ****

On the other hand, a search tree shows both a history of search and the back-up procedures. Thus, it is the search tree that we need to utilize in a complex problem-solving domain. However, it still suffers from the problem of dilute attention.

In sum, it is a nondeterministic characteristic of backward inference that makes learning in a complex problem-solving domain difficult. We need to provide students with scaffolding both for choosing operators and for backing-up at dead-ends. The search tree has potential information to carry out those scaffoldings, but it still remains the problem of dilutes attention. Consequently, the challenge here is how to provide a compact reification of a search tree.

In this paper, we discuss a reification of problem-solving strategies for an advanced geometry theorem proving. We especially focus on teaching construction of auxiliary lines and points, which is one of the most challenging and creative parts of geometry theorem proving. Figure 2 shows an example of the problem requiring construction. In the remaining sections, we first discuss the major issues to teach geometry theorem proving followed by an overview of the previous works. In section 3, we introduce a GUI to reify a search tree generated by GRAMY – an automated theorem prover for auxiliary line problems. Finally, in section 4, we discuss how the reification provides students with better scaffolding.

[1] This should be reffered to as a *solution* tree in general. However, we prefer the word "proof," because we are working on geometry theorem proving.



Given: $BC = 2AB$
D midpoint of BC
E midpoint of BD
Goal: $\angle EAD = \angle DAC$

Figure 2: An example of the problem that needs a construction.

2 Geometry Theorem Proving

2.1 The domain

In geometry theorem proving, the operators to be applied are geometry theorems. A state is defined by a geometric configuration, the known geometric relations (the facts, for short) in the configuration, and a goal(s) to prove. The subjects of state transition are not only the facts and the goal, but also the configuration; the configuration is changed by a construction.

It is well known that, for geometry theorem proving, forward inference works more efficient than backward inference [6]. Indeed, forward inference suppresses the subgoaling by adding known facts without blowing up the inference, because there is no infinite chain of forward inference given each single fact is allowed to be asserted once. However, it is not easy to realize all the applicable theorems at a state, because the problem figure might be complicated enough to hinder the students in reading a necessary configuration off the problem figure.

Backward inference is not necessarily needed to prove theorems that do not require a construction of auxiliary lines. However, it is needed for the theorems that require a construction. Through the experiments with GRAMY, which is introduced in the following section, we have observed that most of the auxiliary lines and points are constructed by backward inference. That is, most constructions are determined by partially overlapping a configuration of known theorem against the problem figure to see the lacking segments. However, it is notoriously difficult for human beings to retrieve an appropriate theorem from their memory to conduct a construction by the partial matching method. So, we need to help students to see how the partial overlapping takes place.

Besides the construction, most importantly for human beings, backward inference helps to think in goal oriented way, and it is more natural and rational way of reasoning for human beings. *** ANY CITATION FOR THIS CLAIM? *** As we discussed above, this requires a backing-up procedure which is not easy for students to maintain.

[[SCHEINES' WORK GIVES ANY SUPPORT HERE?]] [8]

In sum, we need to teach students forward inference to deduce new facts, and backward inference to carry out a construction and to maintain a goal directed reasoning. The latter strategy needs backing-up at an impasse. The main difficulties to teach those topics push us to provide students with scaffolding on the theorem application, the construction, and the backing-up procedure.

2.2 Practical ITSs on Geometry Theorem Proving

Many ITSs and CAIs have been developed for Geometry Theorem Proving so far, yet no one has developed an ITS to teach auxiliary line construction.

Geometry Tutor, one of the earliest geometry ITSs, developed by Anderson *et al.* [2], provides students with a GUI where the students can build up a partial problem tree. That is, they can build a proof by making a connection between the givens placed at the bottom of a screen and the goal placed at the top of the screen. The students can use graphical icons representing the geometric theorems as the connector. They can either hang an icon from a current goal to establish new subgoals, or place an icon to hook the known facts on it to derive a new fact. The former corresponds to a backward reasoning, and the latter corresponds to a forward reasoning. It is expected that the students can perform bi-directional search naturally as the human experts do. Furthermore, Geometry Tutor can provide students with hints on theorem application in terms of the structure of a problem tree. However, since it utilized a problem tree as a model of proof, it could not provide any scaffolding on backing-up, and it has potential flaws discussed in the previous section.

A recent descendant, Angle [3], extends the tutoring capabilities of Geometry Tutor by providing the students with a diagrammatic representation of the theorems, called the diagrammatic configuration schema, instead of the labeled icons used in Geometry Tutor. It actually provides students with a better help in the sense of reifying the theorem applications; students could "see" how to apply theorems. However, it still uses a partial problem tree in the interface with which students conduct a proof.

Several ITSs and CAIs have been developed without explicitly using AND/OR trees. For example, Cabri Geometry [10] provides students with fully manipulations on the problem configuration. Students can change the shape of the configuration while remaining the certain constraints given in the problem. This facilitates students to explore the properties held in the configuration. By changing a coordinate of the apex of an isosceles triangle, for example, the students can see that the apex moves on the perpendicular bisector. Although that kind of scaffolding works fairly well, since they do not directly deal with the problem-solving strategies, they should be placed as the different kind of instructional tools than the one addressed in this paper.

3 A Reification of Strategy for Geometry Theorem Proving

3.1 GRAMY: A Geometry Theorem Prover for Auxiliary Line Problem

[[GIVE A BRIEF INTRODUCTION OF GRAMY HERE]]

3.2 Reification of a Search Tree

To design a GUI that reifies a search tree, we started with showing each state along with the applicable theorems (i.e., DSs), and allow students to turn over the states as if they turn the pages each showing a state. Figure 3 shows the basic idea. The left hand side of the

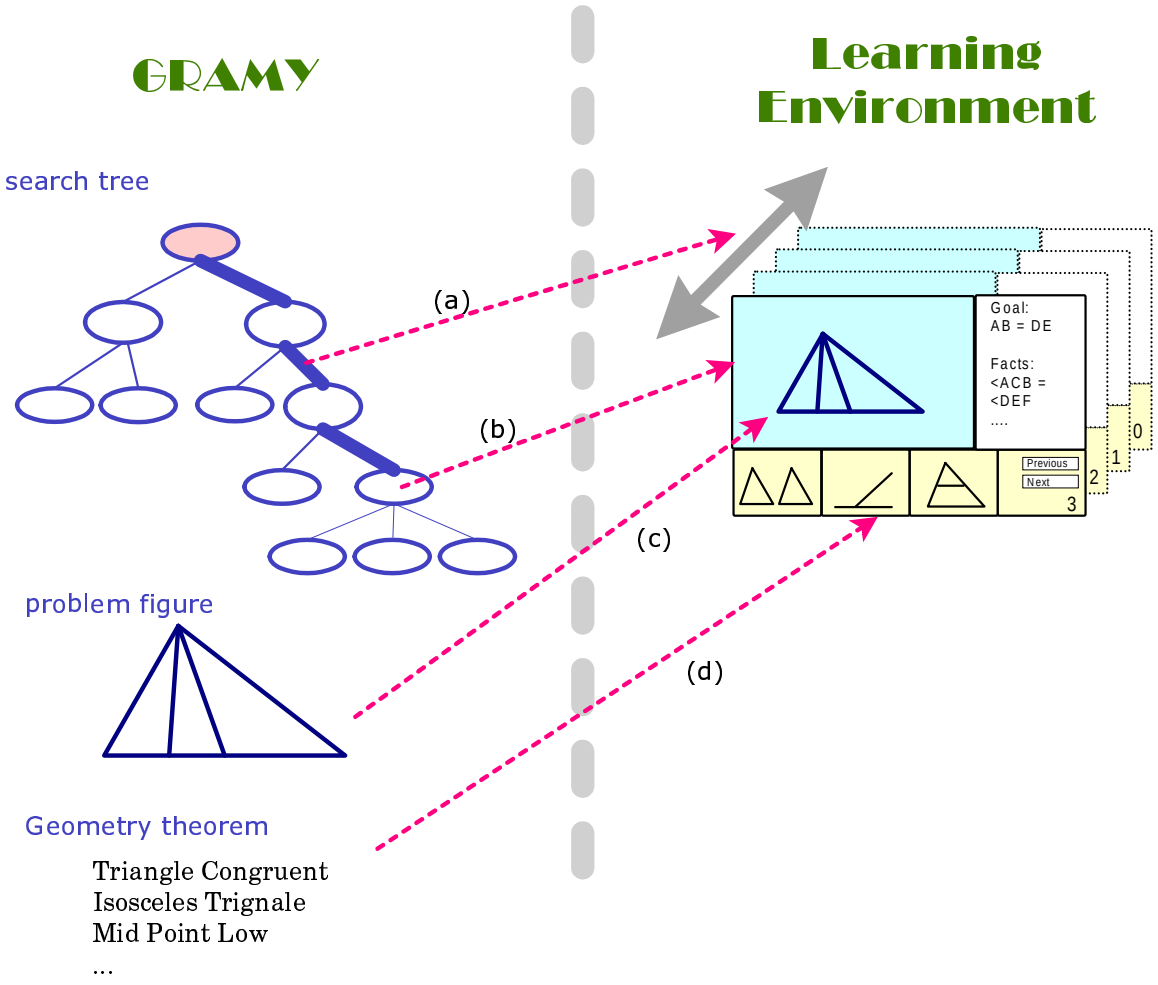


Figure 3: Reified model of geometry proof

figure represents a model of proof in GRAMY. Meanwhile, the right hand side of the figure illustrates a screen shot in a geometry learning environment. Each window (called STATE VIEW) corresponds to a single node in the search tree (indicated by a dotted line `(b)`). The solid window shows the current state and only this window appears on the screen. The other windows drawn by the broken lines, corresponding to ancestor states, are disappeared once turned over. So, the gray thick arrow represents chronological progress of a search (indicated by a dotted line `(a)` associating the pile with several node expansions in the search space specified with thicker links).

Figure 4 is an actual view of STATE VIEW taken from a computer screen. It includes the problem figure, the goal to prove, the facts deduced so far starting with the given premises, and the applicable geometry theorems. By "applicable," we mean either the theorem has a conclusion that matches against the goal (thus, can be used for a backward inference),

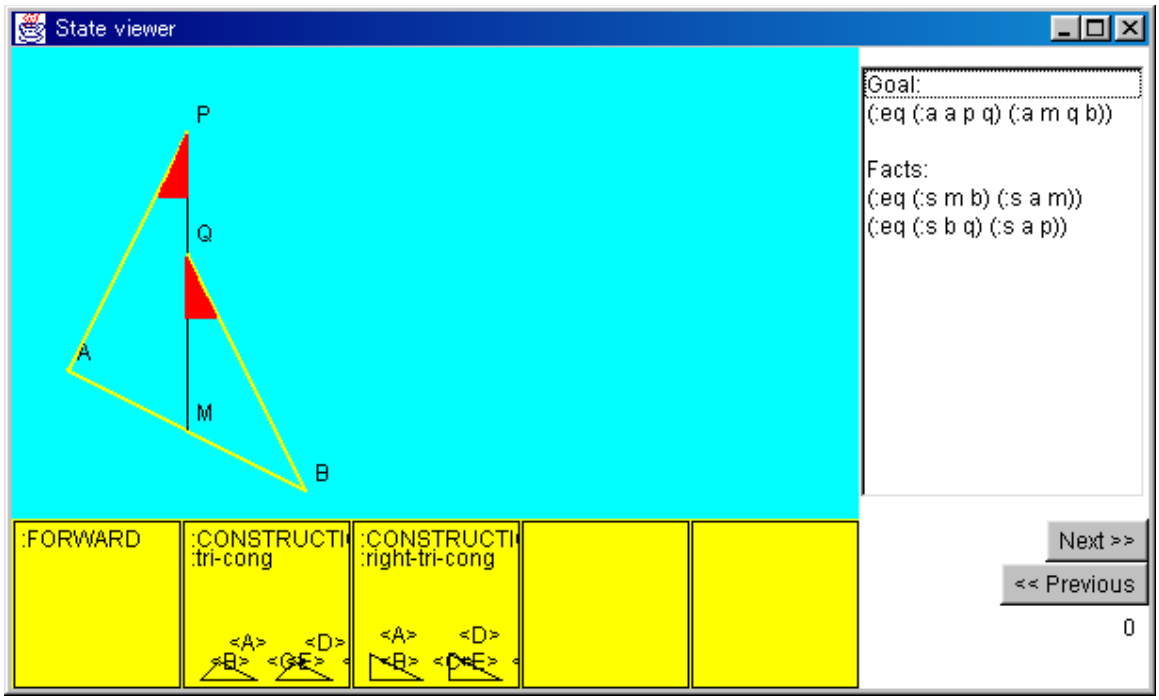


Figure 4: A representation of a proof step.

or it has all the premises satisfied (thus, it immediately derives a new factual statement). The goal to prove and the known facts are highlighted on the problem figure with separate colors. Observe, in Figure 4, the goal $\angle APQ = \angle MQB$, and the two facts $AP = QB$ and $AM = AB$ are all highlighted. When one of the statements (i.e., either the goal or the facts) is clicked, corresponding geometric objects are blinking in the problem figure.

The applicable DSs are lined up with the iconic representation illustrating configuration of corresponding theorem (a dotted line `(d)` in Figure 3). The experiment of GRAMY running on a corpus of non-auxiliary line problems from the literature showed that the average branching factor is less than 4. It supports our design policy of using the iconic buttons. Furthermore, it has also been observed that a number of possible constructions with a particular DS runs up to 40. Thus, instead of lining up all the possible constructions in STATE VIEW, only the icons of DSs used to determine the construction are appeared at the bottom of STATE VIEW. When those icons are clicked, all the constructions determined by the corresponding DS appear in a pop-up window. Figure 5 shows an example of the pop-up window showing possible constructions by the theorem of congruent triangles.

A pop-up window is also used for DSs applied by forward inference. Since GRAMY applies, in forward inferences, all the applicable DSs at once, there might be a number of DSs in those steps. Then, an existence of a forward inference at a state is shown with an icon saying "`FORWARD`." (See Figure 4.)

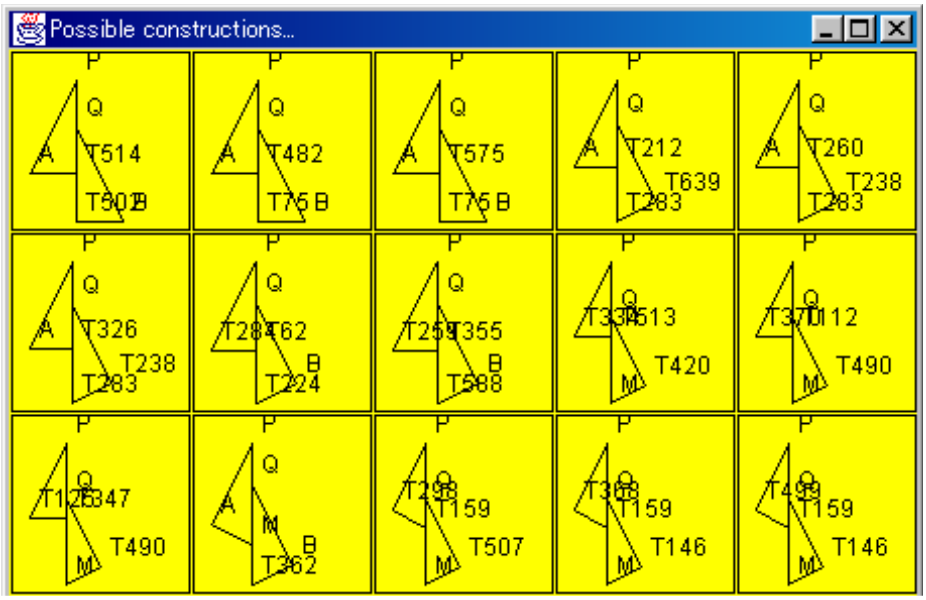


Figure 5: An example of DSs in a pop-up window.

The third kind of icons are for backward inference. They are lined one by one at the bottom of State View.

When an icon in a pop-up window or at the bottom of State View is clicked, a configuration of the corresponding theorem is overlapped against the problem figure, and the theorem is appeared in a separate window, called diagrammatic schema browser (or Ds Browser for short), with which students can browse all the geometry theorems. (See Figure 7.) Ds Browser shows the sentential expression of a theorem together with its configuration. In Figure 6, an application of the theorem of triangle congruent illustlates the auxiliary lines overlapping against the problem figure in State View.

In sum, State View has potential abilities to offer students with the following scaffolding per se.

(S1) Each state in a search tree is reified by visualizing a problem figure, goal to be proven, known facts, and applicable theorems (i.e., DSs).

(S2) The successors expanded at each node in the search tree are reified as the clickable icons.

(S3) An application of a theorem is reified by overlapping a configuration against the problem figure, and also showing sentential expressions in Ds Browser.

(S4) Specific to the auxiliary line problems, but construction is reified as an overlapped configuration against the problem figure.

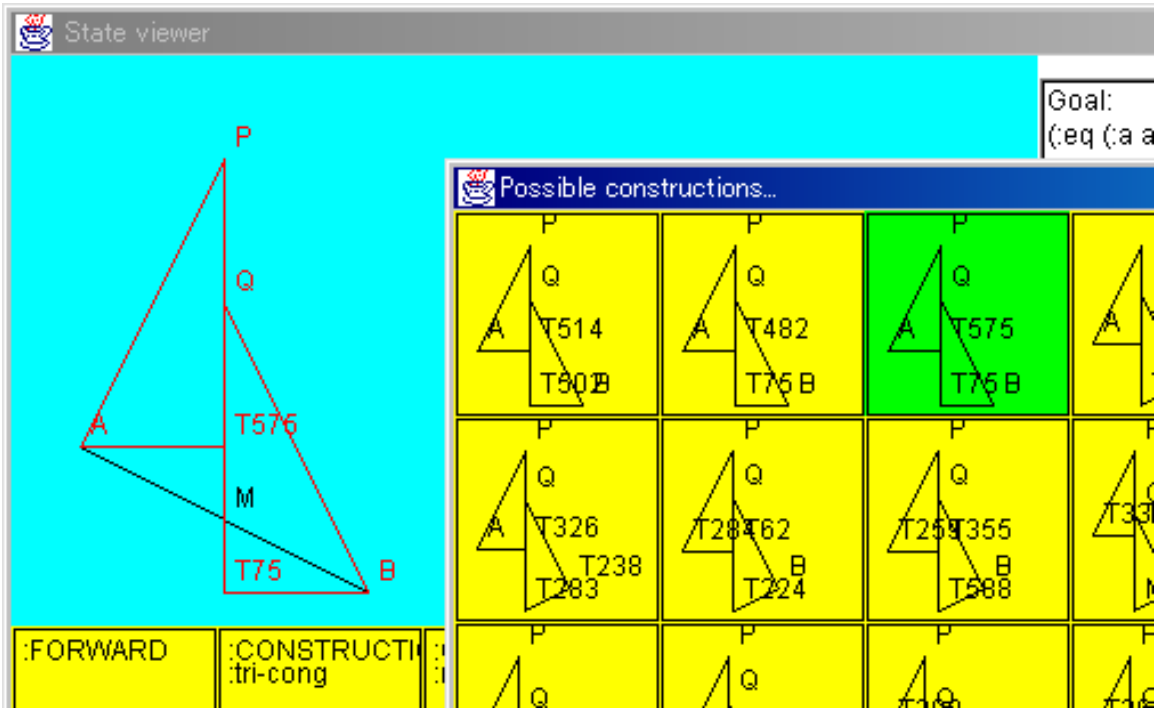


Figure 6: An example of reified DS application.

Finally, when an icon is "selected" by having a double click on it, then the selected icon is highlighted, and a new state is expanded by applying the selected DS to the current state (i.e., where the icon is clicked). Then, the contents of State View are updated accordingly. The figure is modified if the applied theorem is for a construction. The goal is replaced if the theorem is applied in backward manner. The new facts are added if the icon for forward inference is selected.

Now, the students can observe how a proof unfolds in State View by keeping to select an applicable DS (i.e., an icon) at each state. They can return to a previous state by clicking the back button on State View (see `Next` and `Previous` buttons shown in Figure 4). The number at the right-bottom corner of State View shows a depth of the search. This way, the students can freely explore through a search tree back and forth.

Most importantly, the students are allowed to conduct a depth-limited search; that is, they go down a single path in the search tree until they hit a cutoff[2] or a dead-end. When they reach a cutoff or a dead-end, no applicable DS appears so they know that they are at an impasse. At that point, they can "back-up" to a previous state where a selected DS is highlighted, and thus they can take another path.

[2]The cutoff is determined according to the depth of a proof found by GRAMY invoking the iterative-deepening search.

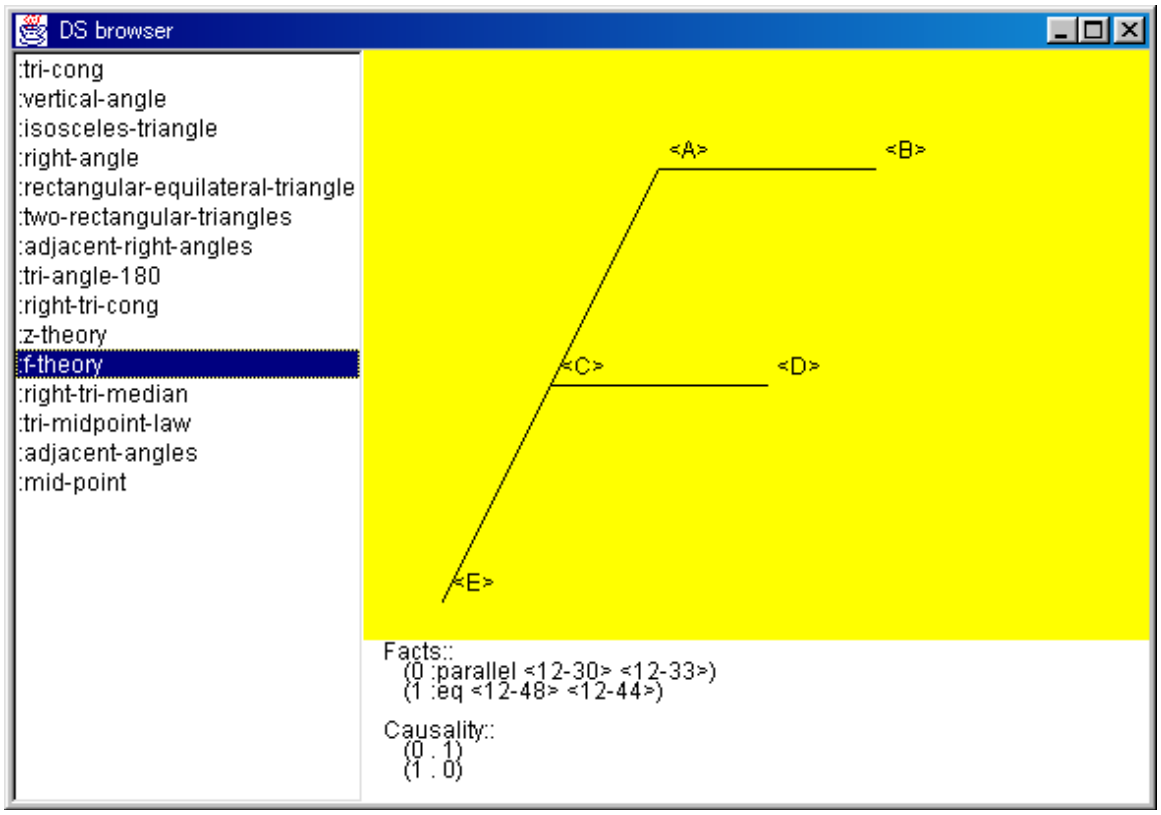


Figure 7: Diagrammatic Schema Browser

In sum, this state exploration facility provides students with powerful scaffolding on learning the problem-solving strategies as follows;

(S5) State expansion is reified as a "selection" of an applicable DS.

(S6) Backing-up is reified as turning over the states in backward way, and selecting an alternative icon.

4 Discussion

Although this research project is at the very beginning stage so that no effort has paid for the experimental evaluation, several expectations of instructional benefit emerge by considering the scaffolding policies listed above.

First, unlike showing a (partial) problem tree or a proof tree, all what reified here is a search tree (S1-S6). Especially, since all the steps necessary for the search algorithm (i.e., depth-limited search) are reified in S1, S2, S5, and S6, it is highly expected that students are facilitated to learn the problem-solving strategies.

Second, since our GUI provides the rationale of a theorem application visually and literally (S3), it is also expected that they could hardly absorb a shallow knowledge of theorem application. A study on forcing students to state a reason for theorem application shows that always exposing the students to the rationale of a theorem application prevents them from learning shallow knowledge [1].

Third, [[HOW WELL DOES THE VISUALIZATION OF THEOREM APPLICATION WORK? ANY IMPLICATION FROM MARSHA'S WORK?]] [4]

Fourth, the students can "envision" a construction without any magic (S4). The powerfulness of GRAMY shows that most of the auxiliary line problems can be solved by the single heuristic, namely, "overlap a known theorem and see which segments are missing." The instructions on auxiliary line construction have been mainly focused on the heuristics to find a possible construction [7]. However, such kind of instruction could be hard to understand, because it does not necessarily help students visualize an actual situation being taught. On the other hand, our approach to construction is fairly simple and would be easy to learn.

Finally, our GUI shows minimal information required by the search algorithm. Thus, students might be able to keep track of the appropriate properties necessary to understand the problem-solving strategies.

5 Conclucion

A reification of the problem-solving strategies in a complex domain is addressed. A GUI which reifies a search tree is implemented and potential abilities of desired scaffolding are discussed. The learning environment is powerful enough so that students can see how the problem-solving strategies are utilized to unfold a proof in a single symple window. A central device in the leraning environment (i.e., STATE VIEW) is implemented with the common GUI design which should be familiare to the students who have used a web browser.

Although we focus on a geometry theorem proving, the reification technique used in this paper is general so that it should be applied to other domains which need backward inference as well (e.g., PROOF TUTOR [8] and ALGEBRA TUTOR [**?**], etc).

References

[1] Vincent Aleven, Kenneth R. Koedinger, H. Colleen Sinclair, and Jaclyn Snyder. Combatting shallow learning in a tutor for geometry problem solving. In Barry P. Goettl, Henry M. Halff, Carol L. Redfield, and Valerie J. Shute, editors, *Proc. of ITS '98*, pages 364–373. Springer, 1998. Lecture Notes in Computer Science, No.1452.

[2] J. R. Anderson, C. F. Boyle, and G. Yost. The geometry tutor. In *Proceedings of the 9th IJCAI*, pages 1–7, 1985.

[3] Kenneth R. Koedinger and John R. Anderson. Reifying implicit planning in geometry: Guidelines for model-based intelligent tutoring system design. In S. Lajoie and S. Derry, editors, *Computers as Cognitive Tools*. Erlbaum, 1993.

[4] Marsha C. Lovett and John R. Anderson. Effects of solving related proofs on memory and transfer in geometry problem solving. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 20(2):366–378, 1994.

[5] Douglas C. Merrill and Brian J. Reiser. Scaffolding effective problem solving strategies in interactive learning environments. In *Proc. of the Annual Conference on the Cognitive Science Society*, pages 629–634. Erlbaum, 1994.

[6] Arthur J. Nevins. Plane geometry theorem proving using forward chaining. *Artificial Intelligence*, 6:1–23, 1975.

[7] George Pólya. *How to Solve it*. Doubleday Anchor Books, NY, anchor books edition, 1957.

[8] Richard Scheines and Wilfried Sieg. Computer environments for proof construction. *Interactive Learning Environments*, 4(2):159–169, 1994.

[9] Mark K. Singley. The reification of goal structures in a calculus tutor: Effects on problem-solving performance. *Interactive Learning Environments*, 1(2):102–123, 1990.

[10] University Joseph Fourier of Grenoble and the CNRS. *CABRI Geometry*. `http://www-cabri.imag.fr/index-e.html`.