Bi-directional search for bugs: A tool for accelerating knowledge acquisition for equation-based tutoring systems

Sung-Young Jung, and Kurt VanLehn
Intelligent System Program
University of Pittsburgh
{chopin|vanlehn}@cs.pitt.edu

Abstract. Authoring the knowledge base for an intelligent tutoring system (ITS) is difficult and time consuming. In many ITS, the knowledge base is used for solving problems, so authoring it is an instance of the notoriously difficult knowledge acquisition problem of expert systems. General tools for knowledge acquisition have shown only limited success, which suggests developing tools that apply only to specific kinds of knowledge bases. Pyrenees is an ITS whose knowledge base is composed mostly of conditioned equations. We have developed several tools for authoring Pyrenees knowledge bases. This paper focuses on a novel and particularly powerful tool that uses bidirectional search to locate bugs in the knowledge base. In several evaluations, human authoring was significantly faster when the tool was available than when it was unavailable

Keywords: Bi-directional search, authoring tools, automatic error detection on knowledge.

1   Introduction

ITS can be classified depending on how their inner loops represent domain knowledge. One classification, which is the focus of this paper, is tutors whose domain knowledge solves the same problems that the students does and thus "model" the desired ways to solve them. These ITS are sometimes called *model-tracing tutors*, although that term is often taken to denote the particular technology used by CMU tutors [8] and Carnegie Learning (http://www.carnegielearning.com/).

   In this paper, we describe the authoring tools used with Pyrenees [10] [12]. Pyrenees uses a large set of knowledge components, called principles, to solve a problem. However, what makes Pyrenees unusual is that most of its principles are conditioned equations. That is, such a principle asserts that under certain conditions, a certain equation is true. This allows a sophisticated error detection method to be used to located buggy principles.

   In Pyrenees, Each principle is represented by a condition and an equation, where the condition indicates when the equation holds (Fig. 1). A set of problems, like the principles, are expressed in terms of the ontology. Fig. 2 illustrates a problem named "isobaric expansion of water" which would be stated in English as "A reservoir contains a liquid, called water, of mass 0.001 kg and heat capacity 4,196 J/(kg*C). The pressure is held constant at 200,000 Pa. As the water is heated, it increases its volume by 0.000001 m^3 and its temperature by 31 C. What is the work done during this time?"

Pa_true(isobaric_expansion(Gas, T)):-
    gas(Gas),
    time_interval(T),
    constant(var(at(pressure(Gas), T))).
	Pa_equation(isobaric_expansion(Gas, T), W=P*Vdiff):-
	    W=var(at(work_done_by(Gas), T)),
	    P=var(at(pressure(Gas), T)),
	    Vdiff=var(at(diff(volume(Gas)), T))).

Fig. 1. Predicates defining a condition (pa_true) and equation (pa_equation).

```
p_definition(isobaric_expansion_of_water,
 [substance(liquid),
  reservoir(liquid, _, _),
  known(var(mass(liquid)), dnum(0.001, kg)),
  %heat capacity
  known(var(heat_capacity(liquid)), dnum(4196, 'J/(kg*C)')),
  known(var(pressure(liquid)), dnum(200000, 'Pa')),
  known(var(diff(volume(liquid))), dnum(0.00000001, 'm^3')),
  known(var(diff(temperature(liquid))), dnum(31, 'C')),
  sought(var(work(liquid)), dnum(0.002, 'J'))          ]).          %answer; 0.002
```

Fig. 2. An example of domain problem.

   Pyrenees solves problems via a version of backward chaining called the Target Variable Strategy [10] [12]. The basic idea is simple: Given a sought quantity, generate an equation that applies to this problem and contains the sought quantity. Include it in the set of equations that comprise the solution.

If the equation has any quantities in it that are neither known nor previously sought, then treat them as sought and recur. When the Target Variable Strategy stops, it has generated a set of equations that are guaranteed to be solvable.

2 Error Detection using bidirectional search

Whenever we add a new problem, we call the problem solver to see if it can be solved. The most frequent sign of a bug is that a problem cannot be solved. This occurs when some principle that should have applied did not get applied. One heuristic to try to find the spot where the principle should have applied is to focus on the dead ends. A dead end is a sought quantity that cannot be calculated from the known quantities. It is likely, but not certain, that a principle should be applied to the dead end, but it failed to apply because the principle was buggy. Although one could examine the dead ends of the tree by hand, there can be hundreds of them.

Bugs that prevent a principle from applying can appear in 3 locations. The bug could be in (1) the principle's condition or (2) in the problem statement. Pyrenees also has a few knowledge components that are now conditioned equations, but instead draw inferences that bridge between a principles condition and the problem statements. (3) Bugs in these rules can also prevent a principle from applying. Fig. 3 illustrates 3 different kinds of errors. These errors are common, hard to notice by inspection and will block a principle from applying.

```
pa_true(isobaric_expanson(Gas, T) ):-          %  Typo. Should be "expansion"
        gas(Gas),
      time_interval(T),
        constant(var(at(pressure(Gas)), T)).     % Wrong parenthesis.  Should be …Gas), T)))


constant(Quantity):-
        known(Quantity, _),
        \+(    Quantity=at(_, T)
      ;    time_interval(T)   ).              % OR (;) should be AND (,)
```

Fig. 3. Examples of bugs

The bi-directional tool starts by creating all possible forward chaining trees. That is, it generates all applicable equations, then use them repeatedly to generate values for all possible quantities. None of the dead end quantities will be among these quantities with known values, because otherwise they would not be dead ends.

However, a dead end may be "one equation away" from the known quantities. Thus, given a dead end quantity, we search for a principle such that one (or more) of its variable specifications unifies with a dead end quantity (*W* in Fig. 4) and all of the remaining variable specifications unify with known quantities (*P* and *Vdiff* in Fig. 4). If we find such a principle, then we know that if it had applied, the dead end would not exist. Thus, a bug must be preventing the principle's condition from unifying with the problems' description.
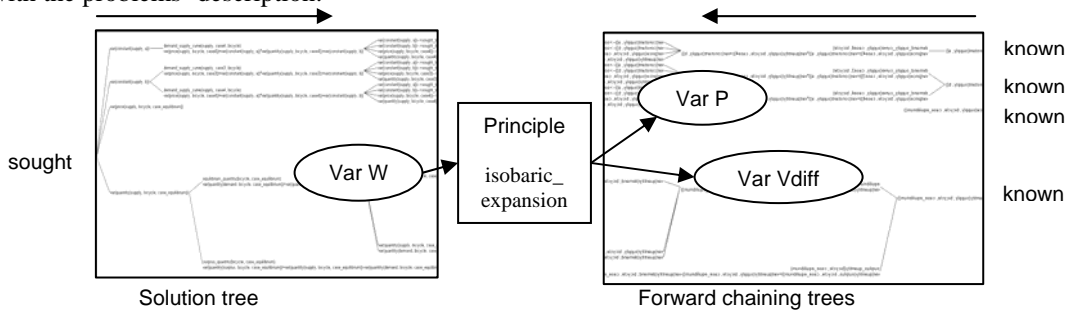

Fig. 4. Bi-directional search identifies a principle whose condition isn't satisfied

3 Evaluation and Conclusions

In order to evaluate the effects of using the bi-directional tool, two kinds of experiments were performed. The task domain was thermodynamics, and it included 14 problems and 15 conditioned-equation principles. The lead author of this paper was the debugger.

Fig. 5-(a) and (b) shows the results for error detection tests on automatically generated errors. The average time spent per a test with the tool was shorter than without the tool (281 vs. 634 seconds, p = 0.00008). Fig. 5-(b) shows debugging time using the tool either the cases when error detection succeeded or failed. The average time spent when error detection succeeded was shorter than when failed (151 vs. 474 seconds). When error detection failed, the time spent didn't show significant difference from the time without using the tool (p = 0.35).

(a) (b) (c)
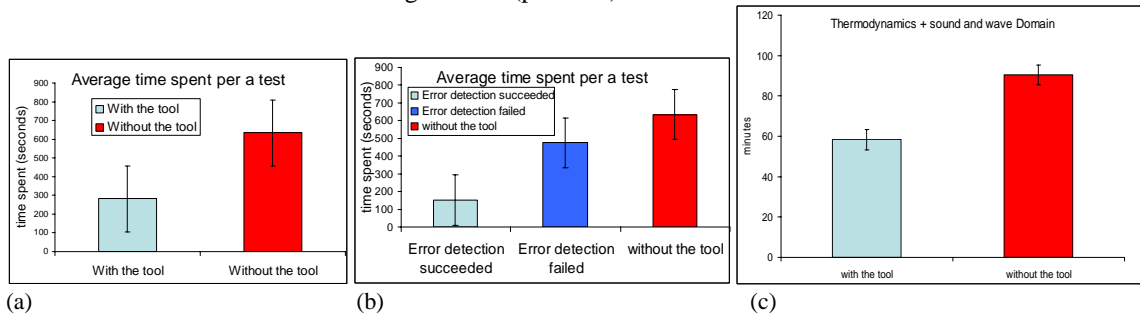
Fig. 5. The result of experiments.

For the authoring test, the number of domain problems was 13 for each domain (total 26). Thermodynamics and sound/wave domains were implemented in each of two passes. Fig. 5-(c) shows the result of knowledge adding tests using the alternation scheme. Authoring the tool was significantly faster than without using the tool (58.38 minutes vs. 90.36 minutes, p =0.017).

Although Pyrenees already had state-of-the-art tools for authoring, including a visualization tool, ontology-based checking of principle semantics and regression testing, we found that a new tool, based on bi-directional search, significantly accelerated authoring. When a problem cannot be solved, the tool searches through the dead ends in the tree generated by the normal, backward chaining problem solver used by Pyrenees. If it can find a dead end that is "one principle away" from a known quantities developed by forward chaining, then it is likely that this principle should have applied but did not because its condition failed to match the problem's description. This localizes the bug, which makes it much easier to spot. Evaluations indicated that the tool saved time, and the difference was statistically reliable.

References

1. Tom Murray, "Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art", International *journal of Artificial Intelligence in Education* (1999), 10, 98-129.
2. Yolanda Gil and Eric Melz, "Explicit Representations of Problem-Solving Strategies to Support Knowledge Acquisition", ISI Technical Report ISI/RR-96-436, 1996.
3. Yost, et. al., "Acquiring Knowledge in Soar", *IEEE EXPERT*, 1993
4. N. F. Noy, W. E. Grosso, M. A. Musen. Knowledge-Acquisition Interfaces for Domain Experts: An Empirical Evaluation of Protege-2000. *Twelfth International Conference on Software Engineering and Knowledge Engineering (SEKE2000)*, Chicago, IL, 2000.
5. MARCELO TALLIS*, JIHIE KIM, and YOLANDA GIL, "User studies of knowledge acquisition tools: methodology and lessons learned", *J. Expt. Theor. Artif. Intell*. 13(2001)35 9±378
6. Yolanda Gil and Jihie Kim, "Interactive Knowledge Acquisition Tools: A Tutoring Perspective," *Proceedings of the 24th Annual Meeting of the Cognitive Science Society (CogSci)*, Fairfax, VA, August 8-10, 2002.
7. Terrence E Turner, K. Koedinger, et. al., "The Assistment Builder: An Analysis of ITS Content Creation Lifecycle", The 19th International FLAIRS Conference, May 11-13, 2006
8. Anderson, J. & Skwarecki, E. (1986). The Automated Tutoring of Introductory Computer Programming. Communications of the ACM, Vol. 29 No. 9. pp. 842-849.
9. Aleven, V., McLaren, B., Sewall, J., & Koedinger, K. R. (2006). The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains. In M. Ikeda, K. Ashley & T.-W. Chan (Eds.), Intelligent Tutoring Systems: 8th International Conference, ITS 2006 (pp. 61-70). Berlin: Springer-Verlag.
10. Chi, M., & VanLehn, K. (2007). Accelerated Future Learning via Explicit Instruction of a Problem Solving Strategy. In K. R. Koedinger, R. Luckin & J. Greer (Eds.), Artificial Intelligence in Education (pp. 409-416). Amsterdam, Netherlands: IOS Press.
11. Koedinger, K. R., Aleven, V., Heffernan, N. T., McLaren, B., & Hockenberry, M. (2004). Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. In J. C. Lester, R. M. Vicari & F. Paraguaca (Eds.), Intelligent Tutoring Systems: 7th International Conference, ITS 2004 (pp. 162-174). Berlin: Springer.
12. VanLehn, K., Bhembe, D., Chi, M., Lynch, C., Schulze, K., Shelby, R., et al. (2004). Implicit vs. explicit learning of strategies in a non-procedural skill. In J. C. Lester, R. M. Vicari & F. Paraguaca (Eds.), Intelligent Tutoring Systems: 7th International Conference (pp. 521-530). Berlin: Spring-Verlag.