RC 17584 (#77588) 1/15/92
Computer Science 14 pages

Research Report

Intelligent Tutoring Systems: Limitations of Current Representations

Ted Selker

IBM Research Division
T. J. Watson Research Center
Yorktown Heights, NY 10598

F.N. Linton

University of Massachusetts
Knowledge Communication
Systems Laboratory
Amherst, MA 01002

IBM Research Division
Almaden • T.J. Watson • Tokyo • Zurich

Intelligent Tutoring Systems: Limitations of Current Representations

Ted Selker
IBM
T.J. Watson Research Center
Selker@watson.ibm.com

F.N. Linton
University of Massachusetts
Knowledge Communication Systems Laboratory
Linton@cs.umass.edu

Number of words: 3200

Short Title: ITSs: Current Representational Limits

Summary:

Idealized Intelligent Tutoring Systems are usually described as having both domain expertise and instructional expertise. Most Intelligent Tutoring Systems, however, have an explicit representation, of only one type of expertise, usually domain expertise. We observe that a few Intelligent Tutoring Systems having separate and explicit representations of both domain and instructional expertise have been constructed and that they are capable of more powerful individualized tutoring than Intelligent Tutoring Systems having an explicit representation of only domain expertise. Furthermore, they are more easily: adapted to new domains, maintained, and evaluated (both domain and instructional modules).

Introduction

In this paper we examine approaches to the design task of constructing separate and explicit representations of domain expertise and tutoring expertise in an Intelligent Tutoring System (ITS). The benefits of including a separate and explicit representation for each of these types of expertise are those of flexibility and modularity; researchers can construct, evaluate, and revise representations of domain and tutoring knowledge more easily if these representations are separate and explicit than would otherwise be the case.

One difficulty that arises when selecting a domain representation for an ITS stems from the fact that procedural expertise, for example, doing subtraction, is more naturally represented in a machine-interpretable form, such as rules, while declarative knowledge i.e., facts, concepts, and principles about subtraction, is more naturally represented in frames of fixed-text. Yet both kinds of knowledge need to be represented if an ITS is to have a full representation of a domain.

A further difficulty that arises when constructing separate and explicit representations of domain expertise and tutoring expertise in an ITS is that the instructional representation must access the domain representation in an instructionally meaningful way. The instructional use of domain knowledge, whether machine-interpretable or fixed-text, is different from the direct application of the domain knowledge (i.e., teaching math is different from doing math, although it includes doing math). Thus the instructional representation must be linked to the domain representation with connections that are meaningful to the instructional representation.

Idealized ITSs are often described as having two expert systems, a domain expert and an instructional expert (Wenger, 1987; Polson & Richardson, 1988). In this idealized model, the expert instructor teaches students the skills of the domain expert by repeatedly comparing students to the domain expert as they solve a sequence of problems, then carrying out any needed instructional activities based on the comparison.

As constructed, however, ITSs tend to emphasize either domain expertise or tutoring expertise rather than the instantiation of both. See, for example, the ITSs described in one recent conference (Frasson, 1988). Systems with machine-interpretable domain expertise (capable of solving problems in the domain) often have little tutoring expertise; furthermore, the tutoring expertise they have is not explicit, but embedded in the code or in the domain representation. In contrast, systems that emphasize tutoring expertise and have runnable tutoring modules tend to represent domain knowledge simply as fixed-text. In general, ITS researchers who actually build ITSs tend to emphasize domain expertise rather than tutoring expertise.

While valid reasons may exist for emphasizing the representation of domain expertise over tutoring expertise or vice versa, the separate and explicit representation of tutoring expertise and domain expertise 1.) allows tutoring expertise and domain expertise to be "mixed and matched," making existing ITSs more readily generalizable; 2.) enhances ease of maintenance, since changes in the design of one area have minimal effect on the other; and 3.) permits the independent evaluation of domain and tutoring modules.

Furthermore, the separate and explicit representation of tutoring expertise - the form of expertise that tends to be neglected - allows researchers to implement, evaluate, and modify tutoring theories, strategies and styles. Finally, a separate and explicit representation of tutoring expertise facilitates the development of a more complex and highly skilled tutor; an ITS that can vary its responses to users on a number of levels, taking into consideration, for example, the nature of the domain knowledge, users' current tasks, their existing knowledge, and their learning styles.

In the remainder of this paper we examine how several ITS researchers have managed the design tradeoff of machine-interpretable vs. fixed-text domain knowledge, and to what extent they have managed to create separate and explicit representations of tutoring expertise and domain expertise. Clancey, in GUIDON, uses a machine-interpretable representation of domain knowledge and a machine-interpretable representation of tutoring knowledge, labeling domain knowledge for instruction by its place in the rule (i.e., he teaches the rules by their parts). Anderson, in Lisp Tutor, uses a machine-interpretable

representation of domain knowledge, attaching an implicit representation of tutoring knowledge (in the form of a tutorial dialogue template) to each rule. Murray, in Statics Tutor, uses a fixed-text representation of domain knowledge, and a separate, explicit representation of tutoring expertise. Domain knowledge is labeled in terms meaningful only for instruction. Selker, in COACH, uses both machine-interpretable and fixed-text representations of domain knowledge, and a separate, explicit representation of tutoring expertise. Each of the design tradeoffs made by these ITS researchers results in an ITS with certain strengths; these are discussed further below. Table 1 summarizes these systems and the representations they utilize.

Example	Domain Representation	Tutoring Representation	Linkage, from Tutoring Representation to Domain Representation
Clancey; GUIDON	D-Rules	T-Rules	T-Rules teach D-Rules.
Anderson; Lisp Tutor	Rules & Textbook	Templates	Templates teach domain rules.
Murray; Statics Tutor	Fixed-text & Simulation	Nets & Rules	Domain fixed-text embedded in tutoring net. No link to simulation.
Selker; COACH	Formal language & Text frames	Rules	Rules select text.

Table 1 Domain and Tutoring Representations and Their Links
Tutors, their main forms of representation of domain and tutoring knowledge, and the predominant form of linkage between the tutoring and domain modules.

To tutor a case, MYCIN first solves the case, then saves the resulting solution trace as a data structure for GUIDON to refer to as it tutors the learner. The domain rules can be run at any time, so MYCIN knows what the student could conclude at any time. A meta-knowledge level keeps track of which conditions of each rule have been satisfied, which subgoals have been met, when a rule has been used, etc. Tutoring rules select a domain rule to discuss, decide what to discuss about it (what part of the rule), and whether to ask a question or provide information. Tutoring rules also respond to a student's input with answers or questions, and update the student model.

Clancey's cleverness in linking tutoring rules to domain rules by rule parts (i.e., the tutoring rules refer to the condition parts, subgoals, and action parts of the domain rules), allows the tutoring rules to refer to the domain rules in an abstract, yet instructionally useful way. This form of linkage, in turn, allowed Clancey to separate tutoring expertise from domain expertise and treat them independently.

Clancey's early success in creating separate, explicit, and linked representations of domain and tutoring knowledge showed the value of this approach and the need for: 1.) representing domain expertise in a form useful for learners, 2.) representing additional domain knowledge as fixed-text, and 3.) representing tutoring expertise in a principled way.

A Seminal Example of Explicit and Separate Representations

A seminal example of separate and explicit representations of domain and tutoring knowledge is found in Clancey (1982). Domain expertise is represented in MYCIN's production system while tutoring expertise is represented in GUIDON's production system. GUIDON uses the case method with interactive dialogue.

The three problems Clancey found with GUIDON have been addressed by later tutors:

- While MYCIN can solve domain problems, the way knowledge is represented does not correspond to a representation of domain knowledge that learners find useful. Anderson, in his Lisp Tutor, described next, represents domain knowledge in a manner suggested by his PUPS cognitive theory, a manner he finds corresponds more closely to learners' needs.
- Domain expert knowledge consists of more information than is found in MYCIN's rules of expertise. There is no place in MYCIN to represent this declarative knowledge as fixed-text. Selker, in his COACH, described below, incorporates a second representation of the domain specifically for fixed-text.
- GUIDON's tutoring rules are ad hoc. Murray, in his Statics Tutor, also described below, presents a more principled formulation of tutoring skills.

An Example Emphasizing Domain Knowledge

Anderson's (Anderson, Boyle, Corbett, & Lewis, 1990) Lisp Tutor has been used with good results in a college level programming course. Anderson's representation of procedural and declarative domain knowledge, tutoring knowledge, and the linkage between them was inspired by Clancey, but is different in several ways. The Lisp Tutor represents procedural domain expertise in production rules written according to Anderson's PUPS cognitive theory. The PUPS cognitive theory distinguishes general-purpose declarative knowledge from use-specific procedural knowledge, and provides a theoretical basis for encoding procedural knowledge in rules, or productions, that are meaningful to *learners* (in contrast to and an improvement over MYCIN which also encoded procedural knowledge in rules, but in a format meaningful to *experts*).

Declarative knowledge about Lisp that Anderson has found to be crucial to learners has been placed, not in the Lisp Tutor, but in an accompanying textbook, *Essential LISP* (Anderson, Corbett, Reiser, 1987). In fact, Anderson has put everything users need to know in the textbook, which, he states, being PUPS based, is "more effective than standard textbooks even without a tutor" (Anderson, et al., 1990, page 14). In any case, this declarative knowledge accompanies the Lisp Tutor, but is not accessible to it for instructional use.

Anderson has embedded instructional expertise throughout his system: some tutoring knowledge is represented explicitly in rules, more is encoded in templates that accompany each Lisp domain rule, and a great deal of tutoring knowledge is included implicitly in the textbook.

Anderson's representation of procedural domain knowledge in a form specifically designed for learners has improved upon the MYCIN representation, which encoded expert knowledge in a form difficult for learners to apprehend. The large amount of declarative domain knowledge in the textbook is evidence that, for comprehensive instruction such as a college level course, a purely procedural representation of domain expertise is not sufficient. Putting the declarative domain knowledge in a textbook, however, makes it inaccessible to the tutor. From another perspective, the textbook and the tutor together comprise a system for teaching that goes well beyond traditional textbooks, since the Lisp Tutor interacts with learners as they write Lisp code, providing learning guidance and feedback that could not otherwise be obtained without a human tutor. The representation of tutoring knowledge has become less explicit and less general than GUIDON's, however, with a corresponding decrease in flexibility and adaptability of the system.

An Example Emphasizing Tutoring Knowledge

Woolf & Murray (1987) provide an example of an ITS that focuses on expert tutoring skills. TUPITS (TUtorial discourse Primitives for Intelligent Tutoring Systems) contains frames with lessons, topics, presentations, and responses; these are selected and presented by TACTNs, (Tutorial Action

Transition Networks) prototypical tutoring strategies and behaviors. Murray has instantiated numerous tutoring strategies in TACTNs. Murray's (1991) physics tutor teaches statics by traversing a statics topic network and presenting the contents of the frames; the tutor selects a frame to visit based on the active TACTN and the kinds of link among the current frame and its related frames.

Murray's tutor uses an explicit machine-interpretable representation of tutoring skills (the TACTNs), and a fixed-text representation of statics in the various slots of the TUPITS frames, but the tutor has no machine-interpretable representation of statics. The system perceives its domain knowledge only as components of instruction. If students want to know the effect of an action in the statics domain (procedural knowledge), they switch to a stand-alone statics simulation where they observe the effects of moving loads and supports, etc., before returning to the tutor.

Murray's tutor illustrates the value of an explicit representation of tutoring skills. Using TACTNs, educational researchers can easily design, implement, evaluate, and modify instructional strategies, which greatly facilitates experimentation. Also, researchers can replace the current domain simulation and domain knowledge with others, and independently evaluate tutoring and domain representations.

A Current Example of Explicit and Separate Representations

Selker's COACH (COgnitive Adaptive Computer Help), is an architecture for teaching syntax and static semantics (Selker, 1989). COACH has been used to teach Lisp programming (Selker & Goroway, 1991) and the UNIX command language. Unlike the tutors described above, COACH does not control the session, but functions unobtrusively as users go about tasks of their own choosing. COACH observes users as they write expressions and displays information about the expressions that users might find helpful. For example, COACH might post a description, an example, or the syntax of an expression. Each of these may be presented at any of four levels of expertise, according to COACH's evaluation of the user's level of skill.

COACH has two representations of domain knowledge. One is a machine-interpretable definition of syntax which COACH refers to in order to evaluate the correctness of a learner's expressions and to prompt the user for the next token. The second domain representation is a frame of syntax-related text associated with each command and token. These frames have slot names that are instructionally significant. For example, the slot labeled **PLUS: Description, Level 1** holds text to be presented to the user when the coaching module infers that the appropriate coaching action is to present a level 1 description of the expression PLUS.

COACH has a rule-based representation of coaching skill. COACH's coaching rules determine, by a consideration of the user's current activity and of the user model, which of the three kinds of help text, if any, to place on the help screen, and which level of help to present. The rules represent coaching heuristics identified by Selker and have descriptive names like Encourage-Exploration and Veto-Arglist. The coaching rules are concerned with the *kind* of domain information in each slot, rather than its meaning; the slot names describe the kind of information in terms meaningful to coaching, e.g., Description, Level 1.

In COACH, domain knowledge is represented in both machine-interpretable and fixed-text formats. Given these, COACH has the ability to interpret learners' actions, and to present them with helpful information as they are working. Representing coaching skill in rules whose conditions refer to the user's activity and to the user model and whose actions refer to instructionally meaningful slot names means that changing coaching styles by adding or revising rules is relatively easy, and that coaching procedures are domain-independent.

Conclusion

Value of explicit domain representation: A machine-interpretable domain representation allows the tutoring module to reason about the domain. Clancey and Anderson's work demonstrates the value of am explicit machine-

interpretable representation of domain expertise: their tutors can demonstrate procedural skills and provide feedback to learners as they practice. Woolf and Murray's work demonstrates the value of a fixed-text representation of domain expertise: their tutors can present information beyond that contained in rules, such as examples, related concepts and principles. Selker's COACH, having domain knowledge represented in both formats, machine-interpretable and fixed-text, has the capability of interpreting learners' actions, and teaching not only machine-interpretable procedural knowledge but related declarative domain knowledge as well.

Value of explicit tutoring representation: Clancey, Murray, and Selker's work demonstrates the value of a separate and explicit machine-interpretable representation of tutorial expertise. Because tutoring and domain representations are separate, domain knowledge can be revised or even exchanged for knowledge from a different domain without affecting the tutorial representation. In other words, the tutorial representation can serve as a shell - different domains can be taught with the same tutor. Because the tutoring representation is explicit, the tutoring strategy can be revised, or exchanged for a different one, simply by rewriting the tutoring rules. These features make it easy to adapt the tutor as the designer, educational researcher, or learner desire.

Value of abstractly-linked representations: Clancey and Selker's work demonstrates the value of knowledge linked abstractly across representations; the links enable the separation of domain and tutoring knowledge, and allow the expert tutor to reason about domain knowledge without being concerned with its content. In Clancey's GUIDON, tutoring rules teach domain rule parts without 'knowing' their contents: the parts of the domain rules serve as labels to link tutoring expertise to domain knowledge. In Selker's COACH, the condition side of the tutoring rules consults the machine-interpretable domain representation in order to determine users' current activity, while the action side of the tutoring rules selects appropriate information from the fixed-text representation for presentation to users. The benefits of these features are summarized in Table 2.

This paper began by noting a discrepancy between idealized ITSs and as-built ITSs: idealized ITSs have explicit representations of domain expertise and tutoring expertise, while as-built ITSs tend to have an explicit representation of only one kind of expertise, usually domain expertise. This discrepancy may perhaps be attributable to the difficulty of representing both declarative and procedural domain knowledge and to the difficulty of finding a suitable linkage between the tutoring representation and the domain representation. Nevertheless, separate, explicit representations of domain and tutoring expertise have several advantages; expertise in each area can be considered independently in terms of:

- selecting the most suitable representation
- modifying or maintaining the representation
- substituting new domain or tutoring expertise
- evaluating performance of individual ITS components
- experimenting with effects of changes in design

Finally, ITSs with separate and explicit representations of domain and tutoring expertise are capable of more powerful tutoring.

Feature:	**Benefit:**
Machine-interpretable domain representation	Demonstrate procedural skills and interactively provide performance feedback to learners.
Fixed-text domain representation	Present information beyond that contained in rules, such as examples, related concepts, and principles.
Explicit tutoring representation	Shell for different domains. Easily modified for changing strategy and experimenting.
Abstract link between tutoring and domain representations	Separation and explicit representation of domain and tutoring knowledge.

Table 2: Features and Benefits of Representations

This table summarizes the features and benefits of explicitly representing domain knowledge, of explicitly representing tutoring knowledge, and of linking domain knowledge to tutoring knowledge abstractly.

References

Anderson, J.R., Boyle, C.F., Corbett, A.T., Lewis, M.W. (1990). Cognitive modeling and intelligent tutoring. In W.J. Clancey & E. Soloway, Eds. *Artificial Intelligence and Learning Environments,* pp. 1-49. Cambridge Massachusetts: MIT Press.

Anderson, J.R., Corbett, A.T., & Reiser, B.J. (1987). *Essential LISP.* Reading, Massachusetts: Addison-Wesley.

Clancey, W.J. (1982). Tutoring rules for guiding a case method dialogue. In D. Sleeman & J.S. Brown,Eds. *Intelligent Tutoring Systems,* pp. 201-225. London: Academic Press.

Frasson, C. (1988). *ITS-88: Intelligent Tutoring Systems.* Montreal, QC, Canada: University of Montreal.

Murray, T. (1991). *A Knowledge Acquisition Framework Facilitating Multiple Tutoring Strategies in an Automated Tutor.* Doctoral Dissertation, Amherst, Massachusetts: University of Massachusetts.

Polson, M.C., and Richardson, J.J., (1988). *Foundations of Intelligent Tutoring Systems.* Hillsdale, New Jersey: Lawrence Erlbaum.

Selker, Ted. (1989). COgnitive Adaptive Computer Help (COACH). In D. Bierman, J. Breuker, & J Sandberg, Eds. *Artificial Intelligence and Education: Proceedings of the 4th International Conference,* pp. 245-251. Amsterdam, Netherlands: IOS.

Selker, Ted., & Goroway, K. (1991). *Demonstrating Usability Improvements in Automatically Presented Adaptive Help* Yorktown Heights, New York: IBM Distribution Services.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems.* Los Altos, California: Morgan Kaufmann.

Woolf, B., & Murray, T. (1987). *A Framework for Representing Tutorial Discourse*. In International Joint Conference in Artificial Intelligence (IJCAI-87). Los Altos, California: Morgan Kaufmann.

Copies may be requested from:

IBM Thomas J. Watson Research Center
Distribution Services F-11 Stormytown
Post Office Box 218
Yorktown Heights, New York 10598