In Proceedings of the ITS-98 Workshop 2 - Pedagogical Agents, San Antonio, TX, 1998.

Architecture of a Pedagogical Agent for Human-Computer Learning and Tutoring

Henry Hamburger and Gheorghe Tecuci

Department of Computer Science, George Mason University
MSN 4A5, 4400 University Dr., Fairfax, VA 22030-4444
{henryh, tecuci}@gmu.edu

Abstract

We propose an architecture for a pedagogical agent (or learning tutor) that can learn from human tutors and then tutor human learners. In addition to its direct practical significance, the endeavor of designing and building such agents provides a conceptual framework for integrating the fields of intelligent tutoring-learning environments (ITLE) and machine learning-based knowledge acquisition (MLKA).

An Approach to Unification

Machine Learning-based Knowledge Acquisition (MLKA) systems and Intelligent Tutoring-Learning Environments (ITLE) are engaged in the same activity except for a role reversal: one system learns from humans and the other helps humans learn. Nevertheless, the two kinds of systems, as well as the fields of Machine Learning and Intelligent Tutoring Systems, have grown up separately and remain largely independent (but see Aimeur and Frasson, 1995).

We have begun to investigate, elucidate and integrate the areas of potential symbiosis of MLKA and ITLE, with a view to facilitating the transfer of knowledge from one field to the other. Our interest is partly at the knowledge level, but we are at least equally concerned with the concrete goal of implementing a unified system with practical benefits. One of us has built a large system, DISCIPLE, that learns (Tecuci and Kodratoff, 1990; Tecuci and Keeling, 1998; Tecuci, 1998). The other of us has built one, FLUENT, that tutors (Hamburger, 1995; Schoelles and Hamburger, 1996). That experience and the actual software produced are contributing to the current work described here. Both projects have put considerable effort into tool-building. In the present work, we continue to build development tools usable by teachers as well as by researchers, enabling teacher-experts to be directly involved in improving educational software.

Our central integrative concept and implementation project is the Learning Tutor (LT), defined as an intelligent agent that can be directly taught by a human and can then tutor human students. Such an agent can serve as an asynchronous communication channel between a human tutor and an unlimited number of human students, one at a time. In the first phase of communication, the agent behaves as an interactive MLKA system, learning directly from a human tutor the domain knowledge and the tutorial knowledge. Then, using the domain and tutorial knowledge, the agent assumes the opposite role, functioning as an ITLE.

A key desideratum for an LT is natural communication. The communication in each of the two phases is natural to the extent that it resembles human tutorial communication. Thus an LT needs natural language processing and a highly flexible two-way graphics communication capability, both functioning under the guidance of a tutorial discourse framework.

The actual construction of an LT drives our comparative study of communication and representation strategies in MLKA and ITLE systems. Both fields provide insights into the appropriate characteristics for a shared internal knowledge representation that serves the operations of learning, tutoring and communicating meaningfully with both human parties, in both linguistic and graphical media. For the representation to be appropriate for communication, it must support the elementary teaching moves of both the human teacher and the agent teacher. Results in human learning and human-computer interaction complement ITLE work as guidance for communication. Results in MLKA are enabling us to overcome the knowledge acquisition bottleneck in the construction of domains, curricula and examples for systems that support human learning.

The next section presents the architecture of the learning tutor. After that we elaborate on the communication issues. The final section is a discussion of our choices of software and chemistry as domains in which the LT operates. We also present a graphical tool, developed by one of us, that enables a non-programmer to build concept-based animations, which are seen to have the potential for a key role in the LT's communicative capabilities.

The Unified Architecture of the Learning Tutor

A key aspect of our strategy of unification is to specify the different kinds of knowledge required in the various components of an ITLE and then to arrange for that knowledge to be directly acquired from a human tutor by MLKA methods. This approach leads to the LT architecture shown here as Figure 1.

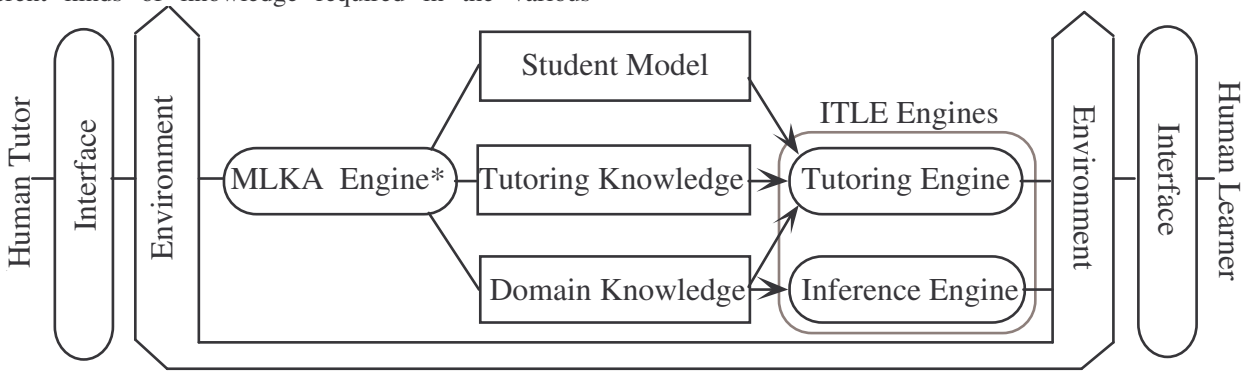

Figure 1 An architecture for the Learning Tutor. Lines indicate two-way flow of information, arrows one way.
*The MLKA Engine has submodules for its various responsibilities as outlined in the text.

Various components of the typical MLKA and ITLE are shared inside the LT as follows. The MLKA Engine, the Domain Knowledge and the Inference Engine constitute a multistrategy apprenticeship learning system, the role of which is to learn the expertise knowledge from the human tutor. Of these three parts, the latter two constitute what is known as the expertise module of an ITLE. Domain knowledge must go beyond performance adequacy; it must be pedagogically appropriate. This is a new demand for LT construction, not previously faced by the independent field of Knowledge Acquisition. Thus our strategy for the problem of obtaining pedagogically appropriate domain knowledge can involve altering the human's stance. The human in this interaction must now be both a tutor and an expert at the same time. This tutor-expert must interact with a view to getting things not only correct but also comprehensible. Just as one communicates knowledge differently to a beginning student than to a colleague, so must the tutor-expert interact differently with an LT than an expert does when imparting a domain to an expert system. During the agent's learning process, the MLKA Engine continuously extends and improves the Domain Knowledge until its domain quality and pedagogical form are both good enough to serve as the expertise for tutoring.

In addition to this more extensive domain knowledge, the system also must acquire explicitly pedagogical knowledge (as opposed to pedagogically appropriate domain knowledge) for the tutorial component. In FLUENT such knowledge takes the form of tutorial schemas provided directly by a teacher-expert working with a tool. In the DISCIPLE approach, the MLKA Engine, the Tutoring Knowledge and the Tutoring Engine in Figure 1 are viewed as a multistrategy apprenticeship learning system the role of which is to acquire tutoring knowledge. The tutor has two top-level subcomponents, one involving curriculum (choosing problems to pose to the student) and one to communicate assistance in ways that are responsive to both the problem state and the student (as represented in the student model). Curriculum knowledge includes such aspects as degrees of difficulty and the prerequisite relationships among the various concepts and problems. It is straightforward to acquire this kind of information. As for communication, the approach is to call upon the tutor-expert for the rationale behind choosing a particular form of communication in a particular situation, by asking, in effect, "explain why you have just given a hint" or "explain why you have just given an explanation." This "why" is not an invitation to a free-form paragraph but is part of a structured dialog using the LT's communication language. Such a scenario involves alternation, within a MLKA session, between acquiring domain knowledge and acquiring pedagogical knowledge.

Yet a third relationship between the two agent roles concerns both domain and tutorial modules, as follows. Each concept or rule from the Domain Knowledge is annotated with information indicating how it has been learned from the human's tutoring moves. The annotations include examples from which the rule or concept has been learned, as well as the various explanations or hints provided by the human tutor. The resulting material then becomes an augmentation to the Tutoring Knowledge of the ITLE. The LT attempts to teach the human learner similarly to the way it was taught by the human tutor, by using these knowledge annotations from the human tutor.

A fourth and final connection is that deploying MLKA techniques within the ITLE not only promotes learning but

also supports the important ITLE function of student modeling. Just as MLKA can build knowledge according to an expert, so too it can build knowledge according to a student, i.e., build a student model. Inferring an accurate student model has long been a main and difficult research goal in Intelligent Tutoring Systems. Our approach of using MLKA to evolve the student model renders this task much more manageable. Indeed, it allows the LT to verify or extend the student model by engaging in a dialog with the student and asking the student what it needs to know, when trying to infer such knowledge would be too difficult. This approach thus stakes out a position that takes into account Self's (1990) maxim - not to guess what the student knows but let her tell you - yet remains responsive to the individual student.

Communication

In order to create natural and compatible communication links among the human tutor, the LT and the human student, all the interactions are based on a common communication repertoire that expresses various types of teaching moves. In DISCIPLE these include providing examples, hints, explanations, problems, definitions, taxonomic and other facts, corrections, and confirmation of partial corrections. These various forms of communication have a spatial as well as a verbal component, notably the diagrams that are so essential to technical communication. In our work, the problem-solving communication between the human tutor or student, on one side, and the LT on the other side, involves knowledge-based, interactive, animated diagrams in addition to textual and symbolic interaction.

Since FLUENT is a conversational system, it too has contributions to make to communication in the LT. Specifically, we are incorporating the notions of views, interaction types and tutorial schemas, described at some length in the work cited above. An interaction type is a short sequence of specified types of linguistic and spatial moves by specified parties. In the Movecaster interaction type, the student makes an action occur and the system describes it. In a Commander interaction the system says what to do, the student tries to do it and the system comments on its comparison of the requested and actual actions.

Linguistic move types within an interaction type include DISCIPLE's communication repertoire mentioned above as well as FLUENT's questions, command and statements. The latter are elaborated by linguistic viewpoints or views, which enable the natural language generation system to take account of some discourse phenomena. They distinguish, for example, between actions and the results of those actions, as in "You picked up the box." vs. "You are (now) holding the box."

Tutorial schemas allow a teacher-expert to organize interaction types including Movecaster, Commander and some others into a coherent educational session, coordinated with meaningful, visible ongoing changes in the underlying microworld situation. The situation is coherent because its activities are guided by plans that make sense in the particular domain. Technically a plan is a flexible tree of subplans and ultimately actions, with constraints on time-sequence where appropriate. The roles in a plan are class-constrained variables, analogous to function arguments. Execution of plans skips any subplans or actions whose goals are currently met. The simplest tutorial schema is just a plan and an interaction type. More generally it can be a sequence of pairs, each consisting of a plan and a regular expression of interaction types.

Two Domains

We have begun to develop the Learning Tutor to learn and teach material in two domains. The first domain is the use of a software tool. The particular software is our own tool for creating concept-based diagrams and animations. Besides providing a domain, this software also supports the learning of other domains, especially technical ones like our second domain, chemistry. Both of these topics are useful, well-understood and problem-oriented. Moreover, the two are distinct enough to push the work toward general applicability.

ITLEs have been most successful in technical education, and we see our work as building on that success. Rapid technology-driven changes in the subject matter of many fields have created a need for educational systems that are rapidly produced and easily updated, yet retain the high pedagogical quality of the best ITLEs. The unified LT approach can provide precisely such a capacity by permitting collaboration with humans in both the learning and the teaching phases. In the long term, we see the evolution of a new generation of educational software, centered around both students and teachers and integrated in its interactions with the two.

Learning to Use Concept-Based Animation

Developed in the context of the FLUENT project, our concept-based graphics and animation tool provides some 100 operations for creating sketches and incorporating them into animations. Even a non-programmer can create animations, define classes and create class instances that inherit the animations of its class. It is also straightforward to specify that certain animated events occur when objects of two particular classes come into contact. A teacher can thus establish an entire microworld that then permits a student to control the combination of events to produce meaningful, continuing situations.

The role of diagram software in a presentation is analogous to that of a human language in an ordinary conversation. To converse meaningfully, you need something to say and a language to say it in. Similarly, to make a presentation with diagrams, including animated ones, you need both domain knowledge and diagram

ability. These two kinds of knowledge are thus complementary. Since it is easier to learn one thing at a time, the learning and teaching by machine and human should proceed counter-clockwise through Figure 2, from the upper left. The left side of the figure has the machine (M) learning a domain from a human teacher-expert (T) in the usual MLKA manner. M's learning to diagram can then be grounded in this new knowledge. Proceeding in a complementary manner, the human student (S) can then acquire diagram skill from the agent in a simple, familiar domain. That knowledge can then support a partly graphical communication in which the human learner acquires new domain knowledge.

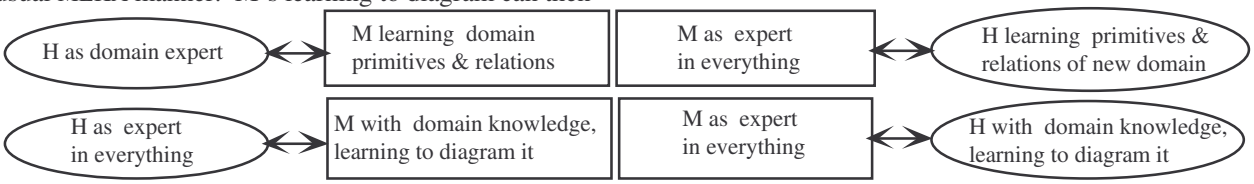

Figure 2 Four kinds of knowledge communication between human (H, in ovals) and machine (M, in boxes)

The following are the key properties of our concept based animation tool:

- There are many operations on points, segments, parts and figures.
- There is symbolic as well as visual control of the graphics.
- Animations include symbolic information along with key frames.
- The graphics and animation are linked to a conceptual representation.
- A builder-user can build animations, classes, objects, concepts and linkages.
- An end user or the system can activate the currently meaningful animations.

Figure 3 shows the major kinds of graphical and conceptual entities that have been implemented and the important relationships between them. As an example of how to interpret the linkages in figure 3, the arrow from FIGURE on the graphical side to OBJECT on the conceptual side means that an object corresponds to the figures that depict it in various states.

The linkage to underlying conceptual representation is crucial to meaningful manipulation of the animated figures and to reasoning about their actions. Together the graphics, animations, knowledge, reasoning and their mutual relationship support the acquisition and communication of knowledge. As an example of the conceptual-graphical link, teachers and learners can: (i) associate graphical constructions to classes, (ii) parameterize the properties of classes, and (iii) create instances of those classes, having particular values for selected parameters.

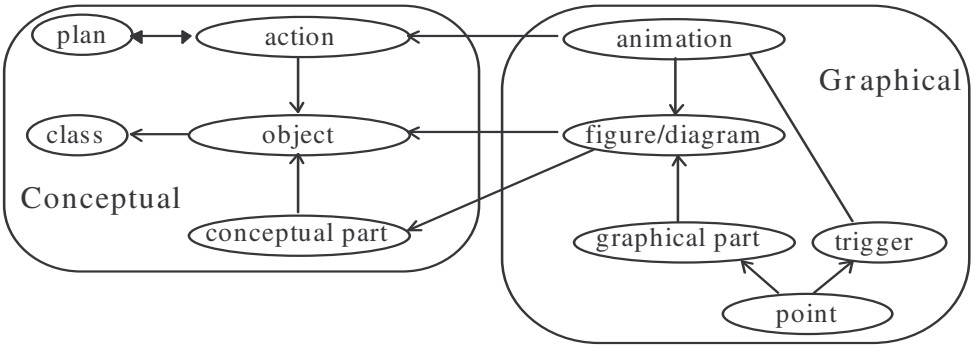

Figure 3 Graphical and conceptual aspects of representation. An arrow means that several instances of the type at its tail may correspond to a head instance. The two-way arrow is a many-many mapping, and the line a one-one mapping.

Chemistry

Chemistry suits our purposes because it has rich, precise knowledge structures involving both symbolic and numerical computation, that are well suited to learning via our knowledge-based, interactive, animated diagrams, which are jointly manipulable by a person and a machine. Some of the Chemistry topics have the additional desirable

property of having two or more different visual representations. There are alternative visualizations for such concepts as element categories and molecule geometry and for various categories of chemical and physical processes. The geometry of molecules, for example, can be represented and communicated as wireframe, ball and stick, or spacefilling models, each with its own advantages for manipulation, envisioning and problem-solving.

Our full-length paper submitted to this conference (Hamburger and Tecuci, 1998) illustrates the use of the LT in the chemistry domain, showing how a chemistry teacher could teach the agent, through examples and explanations, and how the agent could then teach a student, imitating the way in which it was taught by the teacher. We assume that the agent already has some knowledge about chemistry.

To teach the agent how various substances react, the teacher provides a specific example of a chemical reaction (such as $NaOH + HCl \rightarrow H_2O + NaCl$), informing the system about the reaction in several different ways. One way is to interact with the explanation interface of Disciple, explaining to the agent the result of the reaction in terms that are known to the agent, that is, in terms of the concepts and the features from the semantic network. The teacher can also use the drawing tool presented in the preceding section to create a concept-based diagram and to annotate it using free text. That tool can also produce animations.

Goals

Both MLKA and ITLE can benefit from the attempt to transfer knowledge between them. Our strategy is to design a general theoretical framework for the learning and tutoring processes and to refine it as we implement a system that can support learning and tutoring in disparate domains. We aspire to a more comprehensive and unifying view that can contribute to the development of new educational techniques and insight.

Acknowledgments. Part of this research was done in the Learning Agents Laboratory. The research of the Learning Agents Laboratory is supported by the AFOSR grant F49620-97-1-0188, as part of the DARPA's High Performance Knowledge Bases Program, by the DARPA contract N66001-95-D-8653, as part of the Computer-Aided Education and Training Initiative, and by the NSF grant No. CDA-9616478, as part of the program Collaborative Research on Learning Technologies.

References

Aïmeur, E. and Frasson, C. 1995. Eliciting the learning context in co-operative tutoring systems. In Proceedings of the IJCAI-95 Workshop on Modeling Context in Knowledge Representation and Reasoning.

Hamburger, H. 1995. Structuring two-medium dialog for language learning. In M. Holland, J. Kaplan and M. Sams (Eds.) *Intelligent Language Tutors: Balancing Theory and Technology*. Hillsdale, NJ: L. Erlbaum Associates.

Hamburger H. and Tecuci G. 1998. Toward a Unification of Human-Computer Learning and Tutoring. In H.M. Halff and B. Goettl (Eds.) *Intelligent Tutoring Systems: Proceedings of the Fourth International Conference, ITS '98, San Antonio*. Berlin: Springer.

Schoelles, M. and Hamburger, H. 1996. Teacher-usable exercise design tools. In C. Frasson, G. Gauthier and A. Lesgold (Eds.) *Intelligent Tutoring Systems: Proceedings of the Third International Conference, ITS '96, Montreal*. Berlin: Springer.

Self, J.A. 1990. Bypassing the intractable problem of student modeling. In C. Frasson and G. Gauthier (Eds.) *Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education*. Norwood, NJ: Ablex Publ. Co.

Tecuci G. and Kodratoff Y. 1990. Apprenticeship Learning in Imperfect Theory Domains. In Y. Kodratoff and R. S. Michalski (Eds.), *Machine Learning: An Artificial Intelligence Approach,* vol. 3, Morgan Kaufmann, 514-551.

Tecuci G. and Keeling H. 1998. Developing Intelligent Educational Agents with the Disciple Learning Agent Shell. In H.M. Halff and B. Goettl (Eds.) *Intelligent Tutoring Systems: Proceedings of the Fourth International Conference, ITS '98, San Antonio*. Berlin: Springer.

Tecuci G. 1998. *Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies,* Academic Press.