In B.P. Goettl, H.M. Halff, C.L. Redfield, V.J. Shute (Eds.), Intelligent Tutoring Systems, 4th International Conference, ITS'98, San Antonio, Texas, USA, Springer Verlag, 1998.

Toward a Unification of Human-Computer Learning and Tutoring

Henry Hamburger and Gheorghe Tecuci

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

Abstract. We define a learning tutor as being an intelligent agent that learns from human tutors and then tutors human learners. The notion of a learning tutor provides a conceptual framework for integrating the fields of intelligent tutoring-learning environments and machine learning-based knowledge acquisition. We present the conceptual framework of a learning tutor that adapts and integrates major portions of two existing successful systems, one of each kind, originally built separately by the respective authors. It is being developed and applied to the domains of software use and chemistry.

1 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 [1]).

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 ([12], [13], [14]). The other of us has built one, FLUENT, that tutors ([6], [10]). 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 allow to overcome the knowledge acquisition bottleneck in the construction of domains, curricula and examples for systems that support human learning.

The paper is organized as follows. The next section takes note of some relevant developments in MLKA and ITLE and how they influence our approach. Section 3 presents the architecture of the LT. Within that context we then elaborate on the communication issues mentioned above. Section 5 is a discussion of our choices of software and chemistry as domains in which the LT operates, and presents a graphical tool 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.

2 Relevant Developments in MLKA and ITLE

Developments in both MLKA and ITLE make the notion of an LT both timely and promising. We briefly spotlight, in MLKA, apprenticeship learning, multistrategy learning, and programming by demonstration, and, in ITLE, shared control, enriched interfaces and knowledge tuned to tutoring. These developments have clarified the nature of the knowledge and communication that are needed for an effective LT.

Apprenticeship learning is a type of knowledge acquisition and learning in which a learning agent acts as an apprentice to a human expert. The agent assimilates new knowledge from the human expert, by observing and analyzing the way the expert solves problems, as well as by interacting with the expert. ([9], [12], [16]). Multistrategy learning refers to a type of learning in which the learner integrates different learning strategies to take advantage of their complementary nature and to perform learning tasks that are beyond the capabilities of monostrategy learners [8]. Programming by demonstration [5] refers to a type of programming in which a computer system is taught how to perform a task by graphically showing it examples of task performance and explaining them. Disciple embodies elements of all these approaches, allowing a user to build an agent much as one would teach a human apprentice, by giving the agent specific examples of problems and solutions, explanations of these solutions, and supervising the agent as it solves new problems.

One ITLE development important to LT grows from the dialogue on who is in charge. A directive tutor keeps students on the solution path, but a constructive learner learns by wandering in the field of study. Collins and Brown [4] would even wish students to "flounder" in their problem-solving, as a preliminary phase to constructing their own knowledge. Anderson has warned of the potential frustration in floundering, but he agrees that students must learn problem-solving rules by use, not simply by being

told [2]. The enriched interface began in the work of Hollan et al. [7], with their conceptually valid visual representation of a simulation. Generic tools for building such interfaces were later developed by Towne and Munro [15]. Pedagogically tuned knowledge was Clancey's [3] response to a problem that unexpectedly arose when the knowledge base of an expert system was first imported into the expertise component of an ITLE. Knowledge in an expert system is judged by its performance in its domain. When that performance-oriented knowledge is called on to support communication in tutoring, there is no guarantee that it will make sense to novices. The FLUENT system has been directly responsive to the first two of these developments, offering the learner the opportunity to take charge of some parameters that determine system behavior and to relinquish control of others to the electronic tutor. In this way the student can get appropriate guidance without losing ultimate control. Our potential for interfaces arises from the tools described in the section 5.1. Our approach to pedagogical knowledge has been to develop a complex data structure called a tutorial schema, built out of interaction types, linguistics viewpoints and domain plans ([6], [10]).

3 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 Fig. 1.

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

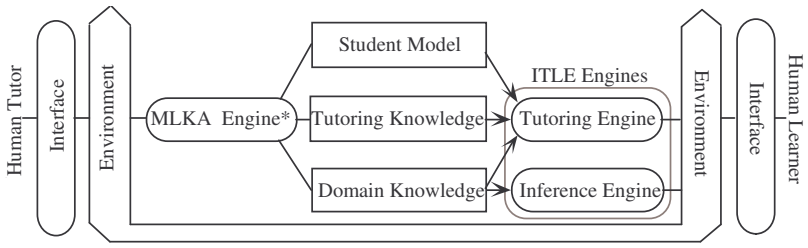


Fig. 1. An architecture for the Learning Tutor. Lines indicate two-way flow of information, arrows one way. *The MLKA Engine has submodules for its various responsibilities as outlined in the text.

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 Fig. 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. This is done via 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 connection is that deploying MLKA techniques within the ITLE supports the important ITLE function of student modeling. 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 like that with the tutor. This approach thus takes into account Self's [11] maxim - not to guess what the student knows but let her tell you - yet remains responsive to the individual student.

4. 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 both a spatial and 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, for instance, the student makes an action occur and the system describes it.

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 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.

5. Two Domains

We have begun to develop the LT to learn and teach material in two domains. The first domain is the use of a software tool, in particular our own tool for creating concept-based diagrams and animations. Besides providing a domain, this tool 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.

5.1 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 present and future importance of learning to use software is clear. Moreover, our particular choice of diagram and animation software supports the learning of an important communication ability. Though often taken to be a verbal skill, communication also has a spatial component – the picture that is worth many words. This is particularly true in technical communication, where diagrams are ubiquitous. Clarity of diagrams is an issue for textbooks, financial reports, appliances manuals and indeed within the software of ITLE systems, where it relates to the important issue of cognitive fidelity of the interface.

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 Fig. 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.

We expect our approach to LT and its application to a drawing tool to contribute to the way people learn to use new software packages. Currently users invest a large amount of time learning a software package. Existing help facilities are based on the idea of letting a user seek needed information, with emphasis on ease of search. However, the user must know what to look for. In contrast, the LT approach is to let the user learn about the software package while trying to solve a specific problem with it. This is possible because the LT has been trained by an instructor by solving problems similar to the user's problem. Therefore, when the user attempts to solve a problem, the LT can suggest a solution, can guide the user through a customized

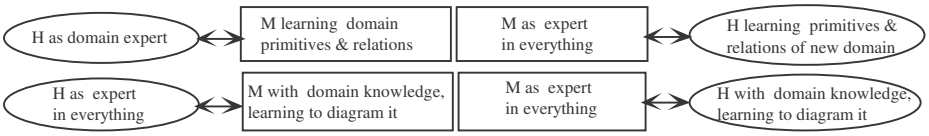


Fig. 2. Four kinds of knowledge communication between human (H, in ovals) and machine (M, in boxes)

manual, or can give advice on how to solve the problem.

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.

Fig. 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 Fig. 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. Similarly, an action may correspond to different animations, depending on the state of its object. As a specific example of one action (on the conceptual side) having more than one animation, the person in Fig. 4 has shapes that face left, right or out from the screen. It can switch among their states by animations for actions of turning and it has other animated actions like walking, which are animated differently depending on which way the person is facing.

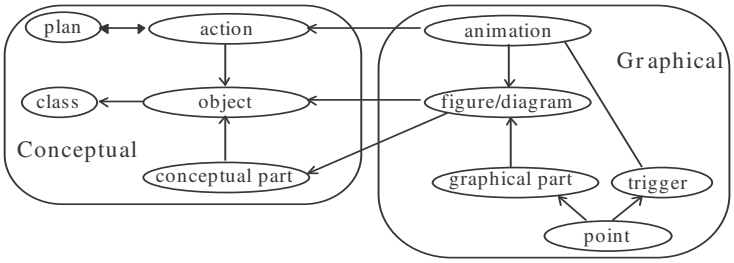


Fig. 3. Graphical and conceptual aspects of representation. An arrow means that several instances of the type at its tail may correspond to a head instance. The two-way arrow is a many-many mapping, and the line a one-one mapping.

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. Another linkage of equal importance is that between teacher and students, via the software. One key to this linkage is the trigger, a region associated with a figure that the user can click to activate a particular action for the object corresponding to that figure. By providing a trigger for an animation, the creator of the diagram enables its user to initiate and control an action that updates the properties of the underlying conceptual object. A teacher can create an animation trigger that students can use to control animations and hence the underlying microworld that they are learning about.

5.2 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.

We now illustrate 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. Let us assume that the agent already has some knowledge about chemical compounds, part of which is represented in the semantic network in Fig. 5.

Fig. 4. An animated microworld

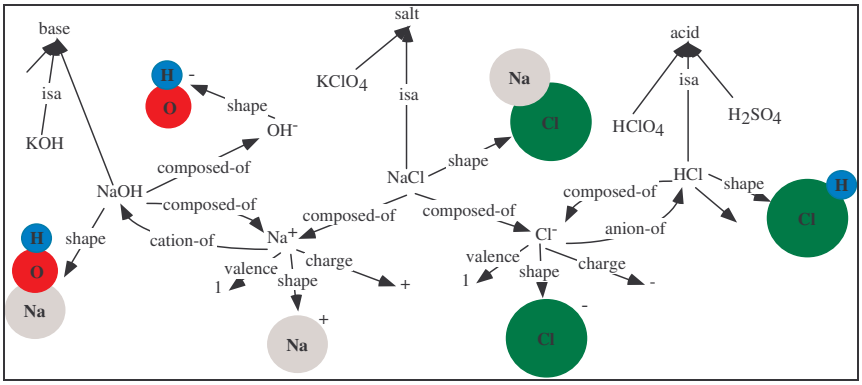


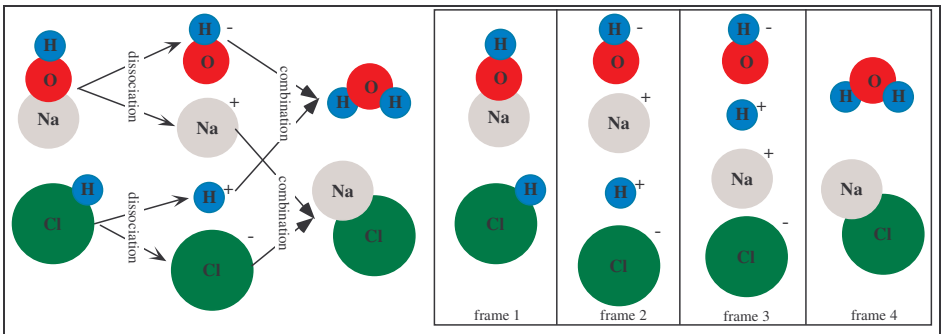
Fig. 5. Part of the agent's knowledge of chemistry

To teach the agent how various substances react, the teacher provides a specific example of a chemical reaction (such as NaOH + HCl → 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 shown in Fig. 5:

NaCl is composed of Na^+ (a cation of NaOH) and of Cl^- (an anion of HCl)

H_2O is composed of OH^- (a component of NaOH) and of H^+ (a component of HCl)

The teacher can also use the drawing tool presented in the preceding section to create a concept-based diagram and an animation, and to annotate them using free text, as exemplified in Fig. 6. The animation of course runs smoothly through interpolations between the frames shown in Fig. 6.



NaOH reacts with HCl to produce NaCl and H_2O. The reaction has two parts:

- NaOH dissociates into OH^- and Na^+, while HCl dissociates into H^+ and Cl^-.
- OH^- combines with H^+ to produce H_2O, while Na^+ combines with Cl^- to produce NaCl.

Fig. 6. Teacher's diagram, animation and the textual annotation of a chemical reaction

At this point, the teacher may give the agent additional examples of reactions (such as, KOH + $HClO_4$ → H_2O + $KClO_4$) that are used to generalize the rule. Or, the agent may use analogical reasoning to hypothesize reactions similar to the one received from the teacher, who will be asked to analyze them. These additional examples (or

counter-examples) of chemical reactions are generalized by the learning tutor, using the knowledge from the semantic network in Fig. 5, to get generalized results like the three listed next, and ultimately to form a general reaction rule, as shown in Fig. 7.

NaOH and KOH are generalized to any base (see the upper left side of Fig. 5)

HCl and $HClO_4$ are generalized to any acid

Na^+ and K^+ are generalized to any metal ion.

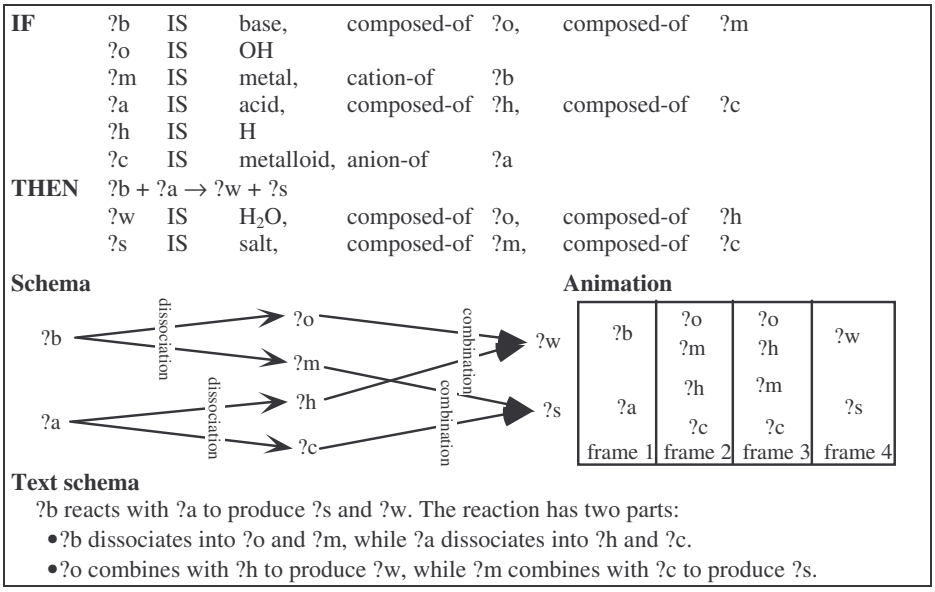


Fig. 7. A general reaction rule

The generalized rule at the top of Fig. 7 states that a base reacts with an acid to produce water and the appropriate salt. This formal representation of the rule allows the agent to perform various types of reasoning but it is not learner friendly. That is, it is an expert-style, performance-oriented rule, not fit for pedagogical purposes. To tutor the student the LT will use rule's annotations. For instance, the agent may imitate, at a conceptual level, the way it was taught by the teacher. For this it is enough to generate instances of the rule in the context of the semantic network in Fig. 5. This will provide examples of chemical reactions (such as, $KOH + H_2SO_4 \rightarrow H_2O + KHSO_4$), illustrated with diagrams and animations similar to those in Fig. 6. Another possible use of the rule is in generating test questions [13] or in verifying chemical reactions specified by the student.

6. 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. Within this long-term goal are these subgoals: 1) Comparative analysis of MLKA and ITLE, with consideration of integration and transfer of techniques, human-machine communication and knowledge representation to support the demands of reasoning, learning and communication. 2) ML-based KA methods for enabling an LT to acquire domain expertise, pedagogical strategies and student models. 3) Prototypes of two LTs.

Acknowledgments: Part of this research was done in the Learning Agents Laboratory which is supported by the AFOSR grant F49620-97-1-0188, as part of the DARPA's HPKB Program, by the DARPA contract N66001-95-D-8653 and by the NSF grant CDA-9616478.

References

1. Aïmeur, E. Frasson, C.: Eliciting the learning context in co-operative tutoring systems. In: Proc. of the IJCAI-95 Workshop on Modeling Context in Knowledge Representation and Reasoning (1995)
2. Anderson, J.R. Pelletier, R.: A development system for model-tracing tutors. In: Birnbaum L.(ed.): *Proc. of the Int. Conf. on the Learning Sciences, Evanston, Illinois*. Charlottesville, VA: Assoc. for the Advancement of Computing in Education (1991)
3. Clancey, W.J.: *Knowledge-Based Tutoring: The GUIDON Program.* MIT Press (1987)
4. Collins, A. Brown, J.S.: The computer as a tool for learning through reflection. In: Mandl H. and Lesgold A. (eds.): *Learning Issues for Intelligent Tutoring Systems.* Berlin: Springer-Verlag (1988)
5. Cypher, A. (editor), *Watch what I do: programming by demonstration*. MIT Press. Cambridge. MA (1993)
6. Hamburger, H.: Structuring two-medium dialog for language learning. In: Holland M., Kaplan J., Sams M. (eds.) *Intelligent Language Tutors: Balancing Theory and Technology*. Hillsdale, NJ: L. Erlbaum Associates (1995)
7. Hollan, J.D., Hutchins, E.L., Weitzman, L.: Steamer: an interactive, inspectable simulation-based training system. *AI Magazine*, 5, (1984) 15-28
8. Michalski, R.S., Tecuci, G. (eds.): *Machine Learning: A Multistrategy Approach Volume 4,* Morgan Kaufmann Publishers, San Mateo, CA (1994)
9. Mitchell T.M., Mahadevan S., Steinberg L.I.: LEAP: A Learning Apprentice System for VLSI Design. In: Kodratoff Y. and Michalski R.S. (eds.): *Machine Learning,* vol III, Morgan Kaufmann, San Mateo (1990)
10. Schoelles, M., Hamburger, H.: Teacher-usable exercise design tools. In: Frasson C., Gauthier G., Lesgold A. (eds.) *Intelligent Tutoring Systems: Proceedings of the Third International Conference, ITS '96, Montreal*. Berlin: Springer (1996)
11. Self, J.A.: Bypassing the intractable problem of student modeling. In: Frasson C., Gauthier G. (eds.): *Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education*. Norwood, NJ: Ablex Publ. Co (1990)
12. Tecuci G., Kodratoff Y.: Apprenticeship Learning in Imperfect Theory Domains. In: Kodratoff Y., Michalski R.S. (eds.): *Machine Learning: An Artificial Intelligence Approach,* vol. 3, Morgan Kaufmann, (1990) 514-551
13. Tecuci G. Keeling H.: Developing Intelligent Educational Agents with the Disciple Learning Agent Shell. In this volume (1998)
14. Tecuci G.: *Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies,* Academic Press, London (1988)
15. Towne, D.M., Munro, A.: Two approaches to simulation composition for training. In: Farr M., Psotka J. (eds.): *Intelligent Instruction by Computer: Theory and Practice*. Washington: Taylor and Francis (1992)

16. Wilkins D.C.: Knowledge Base Refinement as Improving an Incorrect and Incomplete Domain Theory. In: Kodratoff Y., Michalski R.S. (eds.): *Machine Learning: An Artificial Intelligence Approach.* Vol. III. San Mateo, CA: Morgan Kaufmann (1990)