Proactive Learning for Interactive Knowledge Capture

Jihie Kim and Yolanda Gil
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292, U.S.A.
jihie@isi.edu, gil@isi.edu

Abstract

Current tools for interactive knowledge capture have little or no learning aptitude. They are mostly oblivious to the process or strategy that the user may be following in entering new knowledge, unaware of their progress during a session, and ignorant of typical skills expected from a good student. User has to make up for these shortcomings by tracking the status, progress, potential problems, and possible courses of action themselves. We present an approach to make acquisition interfaces more proactive by extending them with: 1) goals that represent what remains to be learned, 2) strategies to achieve these goals and acquire further knowledge, and 3) awareness of the current status of the body of knowledge learned. The resulting interaction shows that the system is aware of its progress towards acquiring the new knowledge, and moves forward by understanding what acquisition goals and strategies to pursue.

Introduction

Acquiring knowledge from end users that have no formal training in computer science remains a challenging task (Blythe *et al.* 2001; Clark *et al.* 2001; Tecuci *et al.* 1999; McGuinness *et al.* 2000; Eriksson *et al.* 1995). Acquisition interfaces use a variety of approaches, including graphical and structured editors, diagnosing errors and helping users to fix them, using existing knowledge to generate guidance for users, etc. However, users are still solely responsible for the acquisition process, in terms of deciding when, what, how, and how well to teach the system. Current technology makes users feel like they are teaching a poor student with a lot of potential, one that has a lot of knowledge but appears to have no interest in learning. As a result, users have to wonder how to assess how well the system is learning new knowledge: is the system acquiring the appropriate kinds of knowledge? did they forget to mention something important? has the system learned enough to answer questions about the material competently? Users that are good teachers will think about these issues and exercise the system accordingly, but they remain responsible for assessing the competence and quality of the system every step of the way. In summary, today's acquisition interfaces can be characterized as passive students with little or no ability to be active participants in the learning process.

The goal of our work is to develop acquisition interfaces that are proactive learners, able to reason about learning activities and with initiative in participating in the process accordingly. Our approach is to enable acquisition tools to have acquisition goals and be aware of the level of competence and confidence of the knowledge they are acquiring. Drawing from limitations shown by previous tools in their interactions with users and from learning/teaching strategies that are typically used in tutoring systems, we formulated goals and strategies to turn acquisition tools into better students. Learning goals have been used to extend automated machine learning approaches successfully (Ram & Leake 1995). In addition, we make our tools aware of the status of what has been learned, i.e., how much they know about a given topic and how confident they are about that knowledge. To show the generality and useful functionality of our approach, we have used it in the context of two different acquisition interfaces with very different input modalities, target knowledge, and testing strategies.

The paper begins discussing the rationale behind our approach as we explored relevant literature in different areas related to interaction and learning. We then introduce our approach and the system we developed in the context of two very different acquisition interfaces.

Proactive Acquisition Interfaces

In order to understand how to build acquisition tools that are more proactive in learning new knowledge from users, we drew from three main sources:

1) Knowledge Acquisition – First, our past experience with users entering knowledge through acquisition interfaces [references omitted for blind review]. By analyzing detailed transcripts of actions taken by users (editing, searching, testing), we could see what knowledge acquisition activities require more effort from users and which actions are more prone to mistakes (some pretty costly to repair). Searching is a typical example of something that is supported by acquisition tools by completely ignoring the context of what the user has been doing, thus making it unnecessarily painful to find relevant items in the KB.

Also, we have consistently seen in user feedback questionnaires many items related to making the system participate more in the process. Many items in these wish lists could be naturally turned into activities that the system should suggest to users when they do not know how to proceed. For example, when the tool points out that there

is an error in the knowledge base the user will understand that there is a problem but not necessarily what to do to resolve it. A very interesting issue that users often bring up is the difficulty of telling whether their actions are helping the system make progress towards acquiring the new knowledge, especially when their changes are making a temporary mess in the knowledge base that only gels together later.

We also analyzed numerous knowledge bases created by users and noticed that they are often incomplete or erroneous in some important way, showing how the tools had failed to help users notice and/or correct these problems. For example, defining the same thing twice under two different names is typical when knowledge entry spans several days. A similar case is defining something that already exists in the KB. Also, defining an item that seems irrelevant to the new body of knowledge, often because users do not realize that they forgot to indicate how it is related to the other knowledge. Despite these errors, the final knowledge bases are still tested for performance. The system shows candidly its logical answer to the questions, with no sense of coverage or confidence based on prior testing. It would seem useful for the system to have some understanding of how much it knows and how well, and use this to qualify its answers.

The users in these studies ranged from experienced knowledge engineers to end users, and the domains included biology, travel, and military planning. The acquisition interfaces and knowledge representation platforms also varied in our analysis. Generalizing across users, domains, and tools helped us formulate patterns of situations and activities where the system could have been much more helpful to the user, by recognizing certain states and triggering appropriate rules containing suggestions for users.

2) Tutoring – Our second source of inspiration is the literature on tutorial dialogues and principles used in intelligent tutoring systems (Forbus & Feltovich 2001; Fox 1993). Their observations helped us understand the level of participation that our tools should aim to have as well as the nature of good teacher-student interactions. We noticed that many useful learning principles could be seen as learning goals and teaching goals that students and teachers seem to pursue at different points throughout a lesson. For example, the topic of the lesson is sometimes presented to the student at the beginning, followed by the content of the lesson, then test questions, and then a summary of the lesson (Woolf & Allen 2000; Fox 1993; Rose *et al.* 2001). Setting up the topic of the lesson at the beginning helps draw on prior knowledge (subsumption to existing cognitive structure) and helps the teacher detect missing prior knowledge that needs to be provided before carrying on with the lesson (Woolf & McDonald 1984; Fox 1993; Ausubel 1968). This would be a useful capability for acquisition interfaces, specially when helping users extend a sizeable body of knowledge. A reasonable expectation in a tutoring situation is that all new items defined must have a connection to the topic of the lesson (Sleeman 1984). Because interactive acquisition tools have no such expectation users often define new knowledge that turns out not to be used in reasoning and perhaps should be (why would the user have bothered to define it otherwise). Testing the student is also a major tutoring activity (Collins & Stevens 1982; Woolf & McDonald 1984; Brown & Burton 1978). Some questions will test the new knowledge with respect to existing knowledge (Rose *et al.* 2001; Ausubel 1968) to ensure it fits adequately. Students should not only be expected to give the right answer but to do so for the right reasons (VanLehn *et al.* 2000). The tutor should be notified if the answer to a question asked previously changes in light of additional material taught (Core, Moore, & Zinn 2000). Acquisition interfaces are not typically very helpful in this sense, and users are solely responsible for the design and thoroughness both test questions and answer analysis. Another interesting aspect of a lesson is learning to describe the new knowledge in terms that are appropriate in the domain at hand (VanLehn *et al.* 2000). Acquisition interfaces rarely focus on acquiring this kind of lexical information, since their goal is to acquire knowledge needed for reasonable performance but not necessarily to give articulate responses. Finally, it appears that it is useful to limit the nesting of lessons to a handful (VanLehn *et al.* 2000), which seems it would help our tools keep track of what is going on as much as it helps a human student.

Many of the tutoring principles suggest a more goal-oriented behavior for acquisition interfaces. Having acquisition goals is key to making a tool truly proactive because it could then steer the dialogue with the user to work towards those goals. Acquisition goals are also useful in order for the tool to understand better what the user is trying to get to.

3) User Interaction Techniques – The third important source of ideas for designing proactive acquisition interfaces is research on various aspects of user interaction, including dialogue planning, intelligent interfaces, and agent-based assistance. Dialogue planning systems guide users in accomplishing tasks by reasoning about plans and goals related to those tasks (Allen *et al.* 2001). The idea of designing a suite of plans for acquisition tasks and planning a dialogue that allows the system and the user to collaborate in solving the task seemed appropriate for building proactive acquisition interfaces. Alas, users have a very wide variety of choices in terms of how to enter knowledge, which results in an unmanageable number of possible acquisition plans that our tools would need to keep track of. Dialogue planning systems work best in relatively narrow domains with relatively few alternative plans. In addition, many discourse reasoning issues handled by dialogue systems (such as reference resolution and discourse obligations) are not central to our problem.

Clearly, some capability to plan acquisition tasks that can accomplish current learning goals is necessary for creating proactive acquisition interfaces. We found ourselves designing acquisition plans that resulted in two extreme interaction modes. They were either too constrained for the user in that he or she had to follow the strategy dictated by the system, or they provided too many abstract and diverse choices in order to give the user flexibility on what to do next. Sensible interaction modes result when systems have heuristics and measures of likelihood that help them avoid making inappropriate or unlikely suggestions (Horvitz 1999).

In order to manage the many possible alternatives in

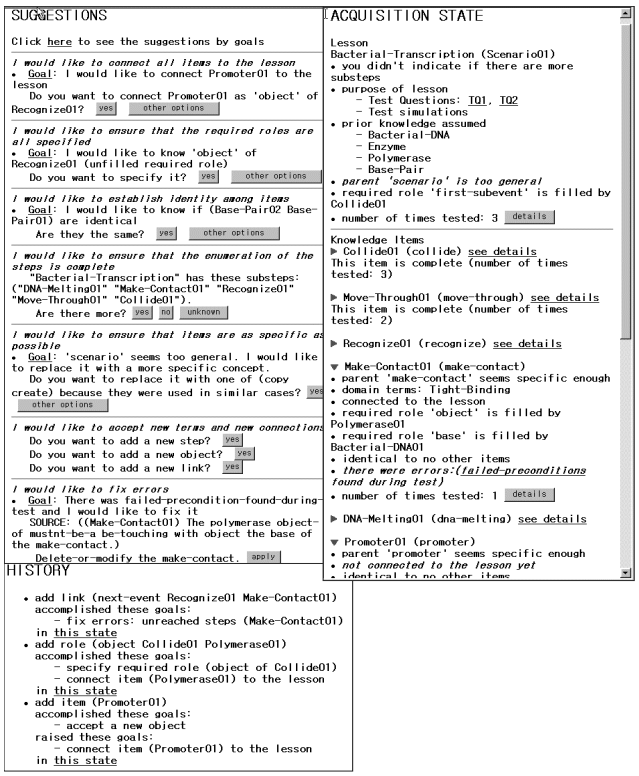

Figure 1: Proactive Acquisition of Process Descriptions.

the dialogue, proactive acquisition interfaces need to generate acquisition plans while reasoning about the priorities of learning goals and tasks, the likelihood of alternative choices, and the appropriate level of detail in their suggestions to the user.

Approach

The key idea behind our approach is to provide acquisition interfaces with the following capabilities:

- Acquisition interfaces should be able to represent explicitly *acquisition goals*. The goals that are achieved at each point during the dialogue represent the progress made towards acquiring the desired body of knowledge.
- Acquisition interfaces should have *acquisition strategies* in order to understand and actively pursue what is involved in learning about a new topic. Acquisition strategies outline how to achieve acquisition goals. Because so many things are unknown to the system during the lesson, these strategies can only be pursued under the user's guidance and in a mixed-initiative interaction.
- Acquisition tools should have *awareness* of what they have learned already and what they do not know about yet, so that they can better assess their competence and confidence in specific topics, and steer the dialogue with the user in directions that improve their body of knowledge on both counts.

We have designed and implemented a system called SLICK[1] that embodies these capabilities. Figures 1 and 2

[1] Skills for Learning to Interactively Capture Knowledge

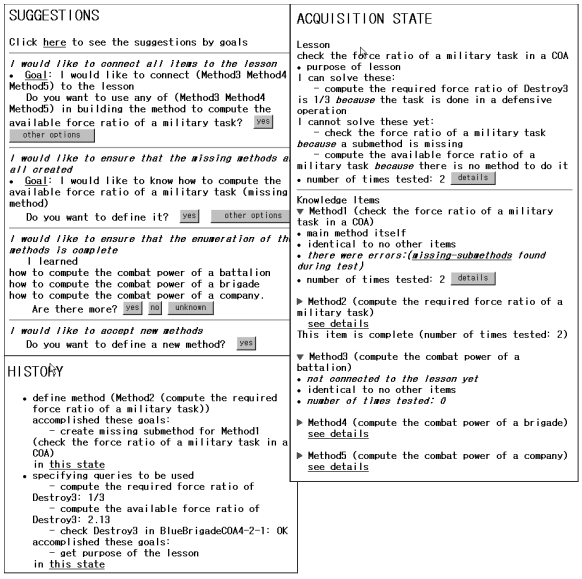

Figure 2: Proactive Acquisition of Problem Solving Knowledge.

show the interface of our system for two different acquisition tools, the details shown will be explained throughout the paper. The left-hand side of the figures show the current set of acquisition goals that are active, and the strategies suggested to the user in order to pursue each goal. The right-hand side of the figures show the state of the knowledge base and the awareness annotations that the system keeps in terms of what is known and unknown and how much the knowledge has been tested. The lower part of the figures show the goals accomplished and raised by every action of the user. SLICK is designed to be layered over the functionality of existing acquisition tools, and uses its own windows (shown in the figures) to interact with the user.

The acquisition tool underlying Figure 1 [references omitted for blind review] acquires process models in terms of their substeps and the objects involved, uses graphical input, and allows users to test the process model by asking questions and by running a simulation. The acquisition tool underlying Figure 2 [references omitted for blind review] helps users specify problem solving knowledge in terms of methods and submethods, uses a structured editor for input, and allows users to pose both parameterized and instantiated problems for testing. This shows that the underlying acquisition tools may be very different in terms of the kinds of target knowledge they capture, the input modality offered to the user, and the testing and error checking strategies used. SLICK makes some general assumptions about the underlying acquisition tool. We consider a new body of knowledge as a collection of *knowledge items* (e.g., actions and objects, problem solving methods or rules, concepts in an ontology editor), each with an associated set of *axioms* (e.g., subgoal relations, range constraints, subclass relations) that embody the knowledge about that item[2].

[2] Axioms are often used in this sense when counting the size of a knowledge base.

- SET UP LESSON AND CHECK BACKGROUND:
  - Get the overall topic and purpose of the lesson.
  - Acquire any assumed prior knowledge before pursuing the lesson.
- ACCEPT AND RELATE NEW DEFINITIONS:
  - Accept new definitions
  - Ensure that new knowledge is specific as possible.
  - Ask the user to be complete when enumerating items in terms of the elements and in terms of the significance of the order given.
  - Get all the information required when existing knowledge indicates it must be provided.
  - Make all new definitions consistent with existing knowledge.
  - Connect all new items with the topic of the lesson.
- TEST AND FIX:
  - Test the new body of knowledge and generate tests for the aspects that have not been thoroughly tested.
  - Fix problems that result from self-checks or from user's indications.
  - Ensure user checks the reason for the answers, not just the answers themselves.
  - Confirm new answers that change in light of new knowledge over what the user had seen the answer to be earlier.
- FIT WITH EXISTING KNOWLEDGE STRUCTURES:
  - Establish identity of new objects by checking if existing objects appear to be the same.
  - Generalize definitions if analogous things exist and there could be plausible generalizations.
- ACHIEVE PROFICIENCY:
  - Acquire domain terms to describe new knowledge.
  - Learn to reason/generate answers efficiently and with shorter explanations.
- REACH CLOSURE:
  - Ensure that the purpose/topics of the lesson were covered and the test questions appropriately answered.

Figure 3: Acquisition Goals.

- For each lesson:
  - purpose of the lesson: overall topic or suite of types of questions
  - background knowledge assumed
  - sub-lessons
  - items and associated axioms
  - overall confidence (based on tests)
- For each item in the lesson:
  - confidence (based on tests over time)
  - connection to the purpose of the lesson
  - relations to other items
  - possible generalizations given existing background k
  - domain terminology details (use default lexical entry, new lexical entries and their contexts of use)
  - inconsistent with (list items or axioms)
  - completeness (item is complete/user has dismissed further questions)
  - identity with respect to other items
  - identity with respect to existing background knowledge
  - analogies with existing background knowledge
  - associated axioms
- For each axiom in an item:
  - completeness
  - generality
  - required information (according to background knowledge)
  - inconsistent with (list items or axioms)
  - confidence (based on tests over time)

Figure 4: Awareness Annotations about Knowledge Acquired.

Users can always revert to the normal acquisition interface of the tool to enter any new knowledge. Actions done by the user through the basic acquisition tool are intercepted by our system. While the backend tool will update the backend knowledge base and its own user interface, SLICK will update its own structures and user interface.

The next sections describe in more detail our approach and its implementation.

Acquisition Goals

Figure 3 shows the general acquisition goals that we use. We found useful to group acquisition goals into six themes, each with a different emphasis on what is being learned. Other goals can be incrementally added in the future.

These high level acquisition goals are mapped to more specific goals to accommodate different acquisition tools and representations. For example, in some cases the purpose of the lesson can be specified as a suite of types of test questions that the system should be able to answer correctly after the lesson. In other cases it could be given as an exhaustive list of new terms to be defined during the lesson. We will discuss this in more detail in the later sections.

Learning Awareness

We represent awareness with two kinds of annotations: annotations to the new body of knowledge acquired and annotations to the interaction history.

A new body of knowledge based is associated with the lesson/purpose/topic of the session(s) where it is acquired. We structure the new body of knowledge as knowledge items and corresponding axioms, as explained earlier. We record this structure (axioms associated with items, items associated with lessons) and extend it as the user goes through the session. This basic structure is annotated with meta-level information about its status, where we aim to capture how much is known about that lesson/item/axiom and how confident the system is about it. Figure 4 shows the annotations that we use.

A novel feature here is the focus on keeping track of what is known, not just on what is not known. Traditionally, the focus of acquisition tools has been on errors and gaps in the knowledge base.

A knowledge base is never complete, so these annotations should ideally become part of the knowledge base or at least in an accessible record of how a body of knowledge was acquired by the system in certain sessions with certain users.

The second kind of awareness annotations that we keep is to the interaction history. They record what action the user took at each point in time (e.g., define a concept as a subclass of another one, define a new role for that concept, test the knowledge with a question), and what progress resulted from that action in terms of the lesson at hand. The system notes the changes to the annotations of the body of knowledge that resulted from the user's action. In addition, the system records what learning goals have been achieved and what learning goals become active, as well as what strategies seem to make sense in order to achieve those goals. These annotations of the interaction history allow the system to share with the user its understanding of what it is learning as the lesson progresses.

Acquisition Strategies

Because acquisition strategies drive the interaction with the user, acquisition interfaces need to strike a balance between exploring and covering all possible strategies that users can follow and not overwhelming them with options that they are unlikely to choose in the first place. This is a very challenging problem and an area of future work, this section describes the approach we currently use.

Consider the situation where a knowledge item is unrelated to all others within the same lesson. This would result in an awareness annotation on that item and activate the acquisition goal of connecting the item. To achieve this goal, the user could pursue all kinds of options, from connecting the item to an existing item, connecting the item to a new item that would be later connected to an existing item,

undoing an existing connection and connecting the item to that, etc. There are many such actions imaginable, each can be instantiated with as many items and axioms exist in the current body of knowledge. Our system currently represents all these choices with an abstraction, namely to connect that item to something, where something has to be instantiated by the user with some item in the knowledge base. Our strategies are single-step strategies, and always correspond to commands that the user can execute through the interface. These commands include to modify a knowledge item or axiom, to ask a question or run a test, and to add a new item or axiom to the current body of knowledge.

The acquisition strategies can be prioritized and instantiated if there is some heuristic or some existing knowledge that apply to the situation. Our system attempts to be a good learner by making educated guesses when possible, and by noting (and possibly exploiting) surprise if its guesses are wrong. If the system can use some heuristics to determine that an instantiation of an acquisition strategy is more likely than others (for example, by drawing an analogy with existing knowledge), then that more concrete strategy would be shown to the user. Priority schemes for acquisition goals also help narrow down which strategies the user may pursue next. As a heuristic, strategies that do not undo previous user actions are considered more likely. Strategies that achieve more than one acquisition goal are considered more likely. For example, a goal to fill in required information of an item and a goal to connect a new item to the lesson can be both solved if the two items are connected (assuming that the first item is already connected to the lesson). Also, we use the six categories of acquisition goals (shown earlier) to order the active goals and present them to the user in that sequence, where the more likely goals are shown at the top. This is because those six categories reflect stages that users typically follow in an acquisition dialogue, although users often jump from one to the other as they see fit.

Proactive Knowledge Capture in SLICK

To show the generality and useful functionality of our system, we present our approach in the context of the two different acquisition interfaces (introduced in Figures 1 and 2 above) with very different input modalities, target knowledge, and testing strategies.

When the user issues a command, the awareness annotations are extended to include its effects. Acquisition goals will be activated, together with the relevant set of acquisition strategies that the user may pursue. Before presenting choices to the user, the system applies heuristics to prioritize the currently active goals and strategies. Our system contains general heuristics that are used by default, but it can also invoke specific modules provided by the underlying acquisition tool and knowledge base reasoners.

Acquiring Process Descriptions

Our first example uses SLICK with a tool that enables users to specify process models in terms of their substeps and the objects involved. Users model a substep as a type of action from the actions available in the knowledge base (e.g., 'Collide', 'Move', etc), and specifies the objects that fill the required roles for that action (e.g., 'agent', 'object', 'location', etc). This tool has undergone extensive user evaluations with expert biologists [citations omitted for blind review]. Figure 1 shows the interface of our system. This particular scenario shows a process model for Bacterial Transcription that combines two different versions built by one of the expert biologists, each contained a subset of the problems that are discussed here and that went unnoticed by the user.

The right-hand side shows the status of the lesson and its knowledge items. The purpose of the lesson is given in this tool as a set of test questions and the overall effects expected after running a simulation of the process. The system also shows the prior knowledge assumed with terms that the user searched in the knowledge base, such as Bacterial-DNA, Enzyme, etc. The knowledge items include actions as substeps (e.g., 'Collide') and objects that play a role in those actions (e.g., 'Promoter'). Inspecting these knowledge items, the user can check their status. 'Promoter' has not been connected to any substep of the process model. The role 'object' of the 'Recognize' step has not been yet assigned.

The learning goals that are active given this state of the lesson are shown in the left-hand side of the screen. For each active learning goal, SLICK shows one suggested strategy (the top-rated one) to the user and offers to show others if the user is interested. The user can ask to see only the learning goals (this view is not shown here), and the system will then show both achieved goals (in this case, for example, setting up the lesson and the background) and remaining ones (in this case, several under the "finalizing new definitions" and "testing and fixing" categories). Here, the goals are ordered according to their priority and the likelihood of the user following the proposed strategy, but notice that the user can examine or pursue any of them. The (heuristic) top level suggestion to the user is to assign 'Promoter' as the 'object' of 'Recognize' since that would accomplish two goals (connecting 'Promoter' to the lesson and assigning an 'object' to 'Recognize'). Some learning goals point to the missing knowledge that was noted in the state. Other learning goals relate individual knowledge items: determining whether two items defined by the user and that have the exact same descriptions are the same ('Base-Pair01' and 'Base-Pair02'), asking the user whether the list of substeps is complete, and ensuring that the parent of 'Bacterial-Transcription' is as specific as possible (its current parent, 'Scenario', is the top-level process in the knowledge base). Finally, some learning goals shown here are the result of testing and fixing errors through a simulation that checks the conditions and effects of each step. In this example, 'Make-Contact' step has a condition to check if its 'object' and 'base' roles are already in contact (because if they already are then it is unnecessary to perform the step).

The bottom of the figure shows how each user action accomplishes and/or raises learning goals. For example, adding a 'next-event' link between 'Recognize' and 'Make-Contact' made the 'Make-Contact' step reachable (in simulating how the steps are executed). The user can also ask to view previous states to further analyze the evolution of the KB.

Acquiring Problem Solving Knowledge

Our second example shows how to use SLICK to guide users to acquire problem solving knowledge. It is shown briefly for lack of space, but with enough detail to illustrate the generality of our approach. The underlying acquisition tool allows users to define individual problem solving methods, each with a method body that can invoke other problem solving methods. The tool has been tested in different domains with a wide range of users, and is shown here with a knowledge base to critique military courses of action that an Army officer was extending.

Figure 2 shows the interface of our system. The right-hand side shows the state of the lesson and the knowledge items. The purpose of the lesson is given as a collection of parameterized problems (to check the force ratio of a military task, to compute required force ratio) and instantiated problems (to compute the required force ratio of Destroy3). Each method is a knowledge item, and is connected to the lesson if it is used to solve a subproblem of one of the problems specified as purposes of the lesson. The learning goals reflect these problems as SLICK tries to help the user decide how to proceed. Unachievable subproblems suggest problem solving methods that remain to be defined by the user. As was the case with the previous system, when a strategy achieves multiple goals it is presented at the top as the most likely one for the user to pursue. Since several methods are not connected to the purpose of the lesson (i.e., not used to check or compute force ratios), and since there is a method still undefined that should achieve one of the subproblems (to compute available force ratio), the system proposes that one of the unconnected methods should be the new method (or part of it).

Conclusions and Future Work

We have presented a new approach for interactive knowledge capture that can be used to extend existing tools with acquisition goals, learning strategies, and awareness annotations over the current state of the knowledge base in terms of its completeness and competence. Our system presents users with useful information regarding the progress made throughout the dialogue, current status of the new body of knowledge, goals that remain to be addressed, and suggested strategies to accomplish those goals. Whether users follow the system's suggested strategies may not be the most important measure of its utility. We believe that the information that the system is capturing about its current knowledge and its progress during the acquisition dialogue gives the user a crucial tool for externalization, i.e., an external record of the teacher/student interaction that helps the user visualize where the lesson is at, relieving users of a significant burden during the acquisition process. The scenarios presented highlight problems in actual knowledge bases that their creators had neither noticed nor fixed.

We will expose our system to users starting this month, in order to gather feedback about what should be shown in the interface. We also plan to extend the work on dialogue plans for acquisition tasks, and incorporate a plan recognition module that relates user commands with multi-step plans. Finally, we would like to incorporate in our system other useful principles of student/teacher interactions. For example, tracking the history to limit the subnesting of lessons and to detect thrashing (defining something, then changing it to fix a problem, then changing it back and getting the problem again).

References

Allen, J.; Byron, D.; Dzikovska, M.; Ferguson, G.; Galescu, L.; and Stent, A. 2001. Towards conversational human-computer interaction. *AI Magazine*.

Ausubel, D. 1968. *Educational psychology: A cognitive approach*. New York: Holt, Rinehart and Winston.

Blythe, J.; Kim, J.; Ramachandran, S.; and Gil, Y. 2001. An integrated environment for knowledge acquisition. In *Proceedings of the IUI-2001*.

Brown, J. S., and Burton, R. R. 1978. Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science* 2:155–191.

Clark, P.; Thompson, J.; Barker, K.; Porter, B.; Chaudhri, V.; Rodriguez, A.; Thomere, J.; Mishra, S.; Gil, Y.; Hayes, P.; and Reichherzer, T. 2001. Knowledge entry as the graphical assembly of components. In *Proceedings of K-CAP-2001*.

Collins, A., and Stevens, A. L. 1982. Goals and strategies of inquiry teachers. *Advances in Instructional Psychology* 2:65–119.

Core, M. G.; Moore, J. D.; and Zinn, C. 2000. Supporting constructive learning with a feedback planner. In *Proceedings of the AAAI Fall Symposium on Building Dialogue Systems for Tutorial Applications.*

Eriksson, H.; Shahar, Y.; Tu, S. W.; Puerta, A. R.; and Musen, M. 1995. Task modeling with reusable problem-solving methods. *Artificial Intelligence* 79:293–326.

Forbus, K., and Feltovich, P., eds. 2001. *Smart Machines in Education*. AAAI press.

Fox, B. 1993. *The Human Tutorial Dialog Project*. Lawrence Erlbaum.

Horvitz, E. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of CHI-99*.

McGuinness, D. L.; Fikes, R.; Rice, J.; and Wilde, S. 2000. An environment for merging and testing large ontologie. In *Proceedings of KR-2000*.

Ram, A., and Leake, D., eds. 1995. *Goal-Driven Learning*. MIT press.

Rose, C. P.; Jordan, P.; Ringenberg, M.; Siler, S.; VanLehn, K.; and Weinstein, A. 2001. Interactive conceptual tutoring in Atlas-Andes. In *Proceedings of AI in Education*.

Sleeman, D. H. 1984. Inferring student models for intelligent computer-aided instruction. In Michalski, R. S.; Carbonell, J. G.; and Mitchell, T. M., eds., *Machine Learning: An Artificial Intelligence Approach*. Springer. 483–510.

Tecuci, G.; Boicu, M.; K, K. W.; Lee, S.; Marcu, D.; and Bowman, M. 1999. An integrated shell and methodology for rapid development of knowledge-based agents. In *Proceedings of AAAI-99*.

VanLehn, K.; Freedman, R.; Pamela, J.; Murray, C.; Osan, R.; Ringenberg, M.; Rose, C.; Schulze, K.; Shelby, R.;

Treacy, D.; Weinstein, A.; and Wintersgill, M. 2000. Fading and deepening: The next steps for Andes and other model-tracing tutors. In *Proceedings of ITS-2000*.

Woolf, B., and Allen, J. 2000. Spoken language tutorial dialogue. In *Proceedings of the AAAI Fall Symposium on Building Dialogue Systems for Tutorial Applications*.

Woolf, B. P., and McDonald, D. D. 1984. Building a computer tutor: Design issues. *IEEE Computer* 17(9):61–73.