A Mixed-Initiative Approach to Teaching Agents to Do Things

Mihai Boicu, Dorin Marcu, Michael Bowman, and Gheorghe Tecuci
Learning Agents Laboratory, Department of Computer Science, MSN 4A5
George Mason University, 4400 University Drive, Fairfax, VA 22030-4444
{mboicu, dmarcu, mbowman3, tecuci}gmu.edu, http://lalab.gmu.edu

Abstract

The research problem addressed in this paper is the development of knowledge bases and knowledge based agents by a subject matter expert that does not have knowledge engineering experience and receives limited support from a knowledge engineer. One possible solution to this problem is provided by the mixed-initiative Disciple approach where an expert teaches a learning agent while solving problems in cooperation. Disciple synergistically integrates the complementary human and automated reasoning to take advantage of their respective knowledge, reasoning styles and computational strengths. The general strategy behind the Disciple approach is to replace the difficult knowledge engineering tasks required to build a knowledge base, tasks that cannot be performed by a subject matter expert, with simpler tasks that can be performed by the expert. The teaching process is illustrated with examples involving the Disciple workaround agent that is able to plan how a convoy of military vehicles can work around obstacles in their path.

1 Introduction

The general research question that we are addressing is the following one:

"How could a subject matter expert, that does not have knowledge engineering experience and receives limited support from a knowledge engineer, teach his knowledge to a learning agent?"

To address this question, we envision a scenario where the subject matter expert teaches the agent how to perform various tasks in a way that resembles how the expert would teach a human apprentice, and the agent learns from the expert, building and refining its knowledge base.

We have addressed this problem for a long time, building a series of increasingly more advanced learning agents from the Disciple family (Tecuci, 1998).

The knowledge base of a Disciple agent consists of an OKBC-type ontology that defines the terms from the application domain (Chaudhri et al. 1998), and a set of plausible task reduction rules expressed with these terms (Boicu et al. 2000). While an ontology is characteristic to a certain domain (such as an ontology of military units, or an ontology of military equipment), the rules are much more specific, corresponding to a certain type of application in that domain (e.g. rules for an agent that assists a military commander in critiquing courses of action, or rules for an agent that assists in planning the repair of damaged bridges or roads).

The general strategy behind the Disciple approach is to replace the difficult knowledge engineering tasks required to build a knowledge base, tasks that cannot be performed by a subject matter expert, with simpler tasks that can be performed by the expert, as shown in Figure 1.

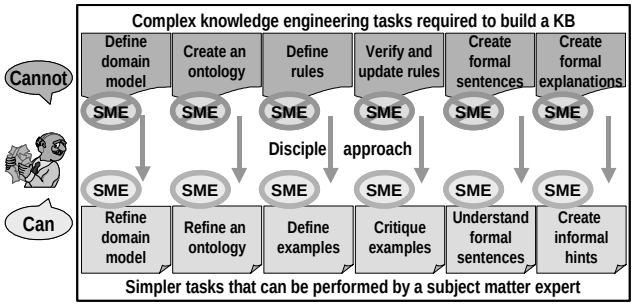

Figure 1: The general strategy behind Disciple

To build a knowledge base, one has first to develop a model of the application domain. Then one has to build an ontology. After that, one has to define problem solving rules, to verify the rules, and to update them. In general, these processes require the creation and modification of formal sentences and of formal explanations.

A subject matter expert cannot perform these tasks. In fact they are very hard even for a knowledge engineer, leading to the well-known knowledge acquisition bottleneck in the creation of knowledge-based agents.

In the Disciple approach, these difficult knowledge engineering tasks are replaced with simpler tasks that can be performed by a subject matter expert, with limited assistance from a knowledge engineer.

Instead of developing a complete model of the application domain, the expert would need to start with an initial model, extending and refining it with the help of the Disciple agent.

The use of the OKBC knowledge model for the Disciple ontology facilitates the import of ontological knowledge from the OKBC compliant knowledge repositories. Therefore, instead of creating an ontology from scratch, the

expert would only need to update and extend an imported ontology (Boicu et al. 1999).

Instead of defining a complex problem solving rule, the expert would only need to define a specific example of a problem solving episode because Disciple will be able to learn a rule from that example. This will be a partially learned IF-THEN task reduction rule. Instead of a single applicability condition, the rule will specify a plausible space of hypotheses for its condition. This plausible version space is represented by a plausible upper bound condition and by a plausible lower bound condition (see Figure 10 for an example of such a rule).

Instead of debugging a complex problem solving rule, the expert would only need to critique specific examples of problem solving episodes and Disciple will accordingly update the corresponding rule.

Most of the time, the expert would not need to create formal sentences or explanations, but only to understand or modify such sentences/explanations that were generated by Disciple, based on informal hints provided by the expert.

2. Mixed-Initiative Reasoning

The Disciple approach is based on several levels of synergism between the expert that has the knowledge to be formalized and the agent that has some knowledge on how to formalize it. This multi-level synergism is achieved through mixed-initiative reasoning that integrates complementary human and automated reasoning to take advantage of their respective knowledge, reasoning styles and computational strengths. The mixed-initiative reasoning is based on a division of responsibility between the expert and the agent for those elements of knowledge engineering for which they have the most aptitude, such that together they form a complete team for the development of the agent's knowledge base. This requires permanent coordination of the dialogue between the human and the agent, and continuous shift of the initiative and control.

At a higher level there is the synergism in solving problems, where the agent contributes routine and innovative solutions, the expert contributes creative solutions, and they cooperate in defining inventive solutions, as indicated in Figure 2.

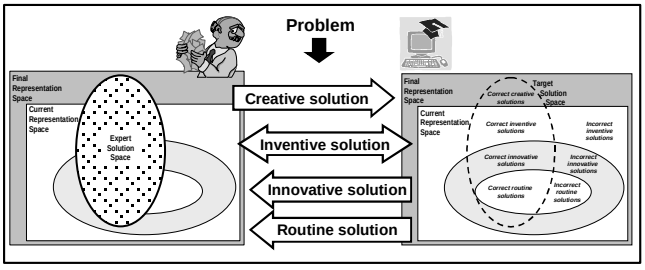

Figure 2: Expert-agent problem solving synergism

The *routine solutions* are those that satisfy the plausible lower bound conditions of the task reduction rules, and are very likely to be correct. The *innovative solutions* are those that satisfy the plausible upper bound conditions. These solutions may or may not be correct and have to be checked by the expert who can accept or reject them. These situations will lead to a refinement of the involved task reduction rules. The *inventive solutions* are based on weaker forms of plausible reasoning (such as partial matching of the plausible conditions of the rules, and tasks similarity based on the structure of the ontology). An inventive solution is based on an analysis of several rules, and is generally a novel task reduction. From inventive solutions the agent will learn new plausible task reduction rules. Finally, the *creative solutions* are those that cannot even be expressed in the current agent's representation language. These solutions have to be provided by expert. They will lead both to an extension of the ontology, and to the learning of new rules. As a result of this learning process, the problem solving situations that were innovative for the agent gradually become routine, and those that were creative, gradually become inventive, then innovative and ultimately routine.

At the next level down, there is the synergism between teaching and learning, illustrated in Figure 3.

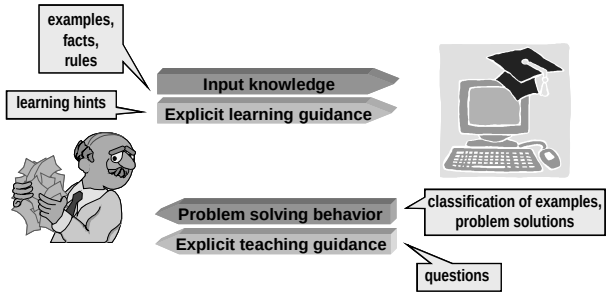

Figure 3: Expert-agent teaching/learning synergism

Generally, the expert provides an input to the agent, in the form of examples, facts, or abstract knowledge. The agent learns concepts or rules from this input and applies them to classify new examples or to solve some problems. The expert analyzes this problem solving behavior of the agent and provides feedback based on which the agent can refine its knowledge. There is, however, another possible feedback loop that could significantly enhance this mixed-initiative teaching and learning process. The expert can help the agent to learn by providing useful hints, and the agent can help the expert to teach it, by asking specific questions.

Besides the synergism between the expert and the agent, there is also the synergism between the different agent's components or employed methods. When the agent learns from the expert, there is the synergism between different learning strategies employed by the agent in situations in which no single strategy learning method would be

sufficient. These methods include learning from examples, learning from explanations, learning by analogy and learning by experimentation (Tecuci 1998). Another example of such synergism is that between domain modeling, problem solving and learning that will be discussed below.

A central component of the Disciple architecture is the mixed-initiative manager that is capable to control the transfer of initiative between different tasks performed by the agent or by the expert, and to notify one another about the relevant changes.

The tasks performed by the user and by the agent are described in a formal language based on a task-decomposition / task composition approach that provides a useful similarity with the general problem solving approach of Disciple. This framework allows the agent to reason about the current tasks performed and to dynamically adapt to the user's actions.

The interactions between the expert and the agent are complex and cannot be entirely predicted and specified during development. The mixed-initiative manager may guide the process toward the task completion by analyzing and providing feedback related to the tasks performed, the tasks currently executed, and the tasks that will need to follow. The User-task Agenda, a key component of the user–agent interaction manager, will exhibit all these behaviors. It will display all the tasks relevant to the user at different levels of abstraction, providing an increasing efficiency of interaction as the human expert becomes more experienced in collaborating with the agent. It will also highlight the tasks that are currently being executed and those that can be scheduled for execution at that time to keep the expert informed with both the overall task and the current status of the execution. Another capability of the User-task Agenda is anticipating expert's intentions by generating all the possible tasks to be executed next, selecting and displaying the ones that correspond most closely to his actions, as a hierarchy reflecting their natural order and inter-dependencies.

The mixed initiative manager also control the proactive behavior of the agent that executes support tasks in advance, in order to facilitate the initiation or completion of an important task that will follow.

Domain modeling, problem solving and learning are the main processes involved during the training of the Disciple agent. The most difficult of them is the domain modeling, which is the process of making explicit, at an abstract and informal level, the way the expert solves problems. Currently, domain modeling is manually done by a knowledge engineer, being the most difficult barrier in enabling a subject matter expert to directly teach an agent. We are therefore investigating ways in which the agent can assist the expert in the domain modeling process, partially automating it. Besides relying on a mixed-initiative approach, we are also researching the possible relationships between domain modeling, problem solving and learning, trying to find ways in which they could support each other, as indicated in Figure 4.

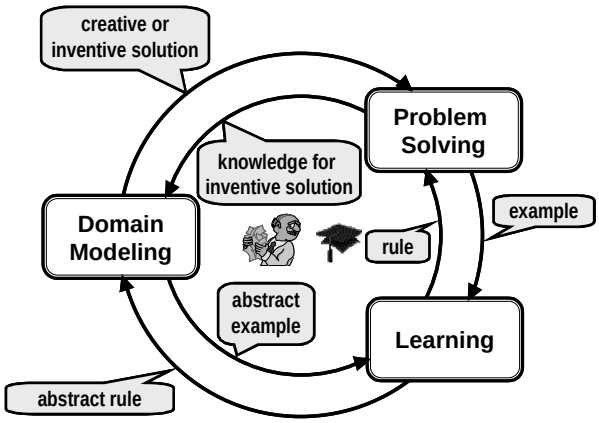

Figure 4: Synergism between domain modeling, problem solving and learning

There is a clear relationship between problem solving and learning. Problem solving provides examples for learning. Learning generalizes the examples to rules or refines existing rules that are used in future problem solving processes.

We are investigating the relationship between domain modeling and problem solving. Both represent the same reasoning process, but at different levels of abstraction and formalization. The domain modeling will provide and refine the models that will guide the cooperative problem solving process. Also a specific problem-solving episode may refine and develop its initial model. When the problem solver cannot solve a problem, the expert needs to provide a creative solution, by first defining an abstract model of it, and then by refining it for the problem solver. There are, however, situations where the necessary solution is not entirely creative, but has some similarities with routine or innovative solutions. In such a case the problem solver can assist the expert in domain modeling by suggesting elements of an inventive solution.

We also plan to investigate the relationship between domain modeling and learning. The knowledge in domain modeling is mostly informal and the agent has only a partial understanding and formalization of it. Therefore the learning algorithms must be adapted to learn general domain modeling rules from these abstract and informal domain modeling examples. These abstract rules may guide and help the user in further domain modeling. However, these domain modeling rules may also help in learning the problem solving rules.

At the basis of this integration and mutual support of domain modeling, problem solving and learning, should be a learnable representation of knowledge that would represent knowledge at various levels of abstraction and formalization: abstract and formal for domain modeling, formal for problem solving and learning, and natural for expert-agent communication.

3. The workaround challenge problem

Over the years, the Disciple approach has been developed and scaled-up continuously, more recently as part of the 1997-1999 High Performance Knowledge Bases (HPKB) program supported by DARPA and AFOSR (Cohen et al., 1998). The organizations participating in HPKB were given the challenge of rapidly developing and updating knowledge-based systems for solving specially designed challenge problems. The aim was to test the claim that, with the latest Artificial Intelligence technology, large knowledge bases can be built and updated quickly and efficiently.

One challenge problem for the first part of the HPKB program was to build a knowledge-based workaround agent that is able to plan how a convoy of military vehicles can "work around" (i.e. circumvent or overcome) obstacles in their path, such as damaged bridges or minefields (Jones, 1998). The input to the agent includes two elements: (1) a description of the damage (e.g. a bridge is destroyed – see Figure 5), and of the terrain (e.g. the soil type, the slopes of the river's banks, the river's speed, depth and width), (2) a detailed description of the resources in the area that could be used to repair the damage (e.g. a description of the engineering assets of the military unit that has to workaround the damage, as well as the descriptions of other military units in the area that could provide additional resources).

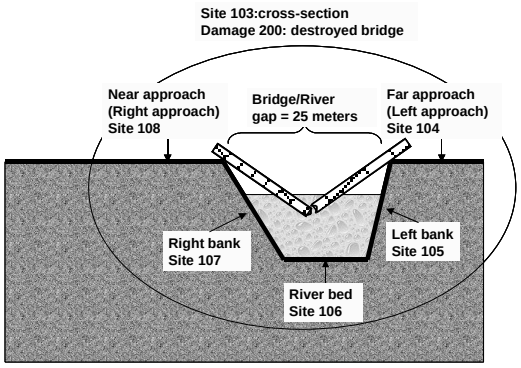

Figure 5: An example of a damaged bridge

The output of the agent consists of the most likely repair strategies, each described in terms of three elements: (1) a reconstitution schedule, giving the transportation capacity of the damaged link (bridge, road or tunnel), as a function of time, including both a minimum time and an expected time; (2) a partially ordered plan of engineering actions to perform the repair, and the minimum as well as the expected time that each of these actions require; and (3) a set of required resources for the entire plan and for each action. To solve this problem we have developed the Disciple-Workaround learning agent (Tecuci et al., 1999).

4. The Disciple agent training process

We will briefly illustrate the process of teaching the Disciple-Workaround agent. The expert starts with a specific task (e.g. "Workaround obstacle by unit10" in Figure 6). Then he asks a question related to this task, the answer of which leads to the reduction of this task to a simpler one (or, in other cases, to several simpler tasks). This process continues until the top level task is reduced to a partially ordered sequence of elementary tasks that represent the plan to workaround the obstacle.

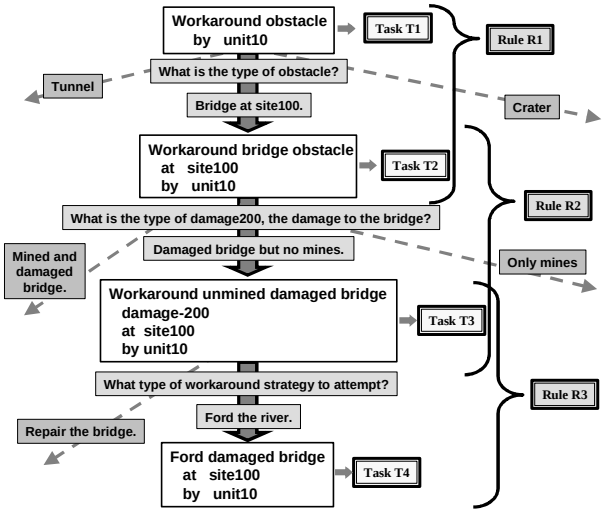

Figure 6: Sample task reduction tree

From each task example Disciple learns a general task pattern. Also from each task reduction step Disciple learns a general task reduction rule. Let us consider the task-reduction example from Figure 7. First Disciple learns general tasks from the task examples in the task reduction, as illustrated in Figure 8. Disciple learns both an internal representation and a natural language pattern for the task. Notice that the ranges of the task features are only partially learned, being represented as plausible version spaces. For instance, the set of possible values of ?O1 could be {site107} or the set of instances of any subconcept of GEOGRAPHICAL-REGION.

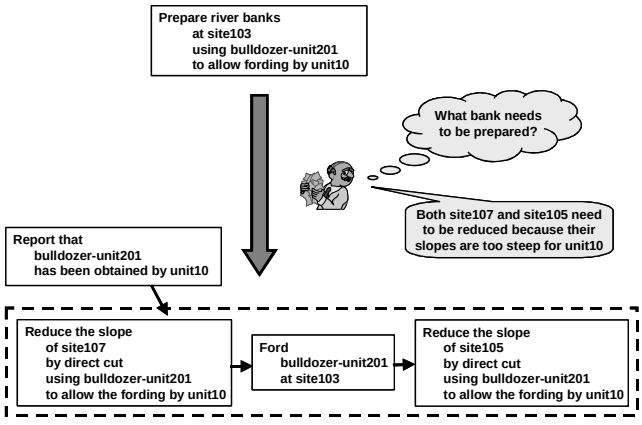

Figure 7: An example of task reduction

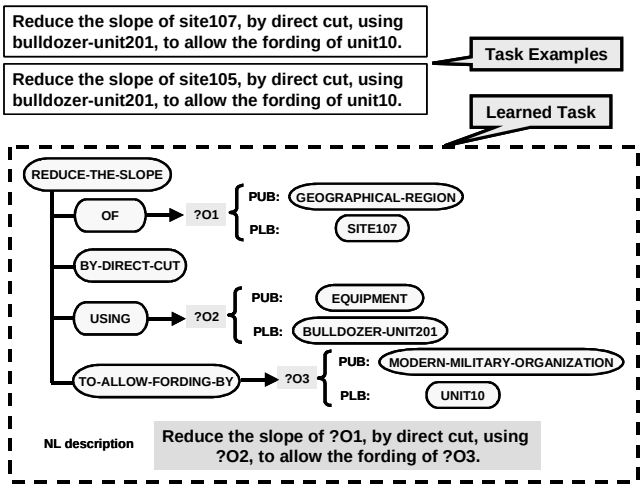

Figure 8: Learning a task description from examples

Disciple also learns a general task reduction rule from this task reduction example. The process of learning the rule is represented in Figure 9. Disciple tries first to understand why the task reduction step is correct. It uses analogical reasoning, natural language processing and hints from the expert to propose plausible explanations from which the expert selects the correct ones.

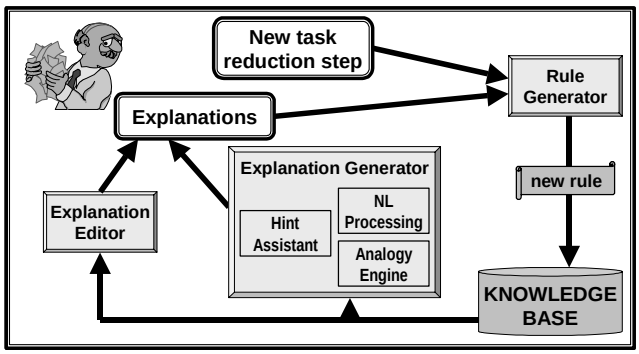

Figure 9: The rule learning process

The explanations of the task reduction from Figure 7 are the following ones:

1. **Unit10 located-at site108** *and*
   Site103 right-approach site108 *and*
   Site103 right-bank site107
2. **Unit10** *has a* **default-negotiable-slope** *of* **25** *and*
   Site107 *has a* **max-slope** *of* **200 > 25.**
3. **Site107** *is on the* **opposite-side** *of* **site105.**
4. **Site105 is** *a* **bank.**
5. **Unit10** *has a* **default-negotiable-slope** *of* **25** *and*
   Site105 *has a* **max-slope** *of* **200 > 25.**

Based on the example from Figure 7 and on the above explanations Disciple automatically generates the complex task reduction rule from Figure 10

The Disciple-Workaround agent was developed very rapidly, based on sample solutions provided by a subject matter expert. During the 17 days of DARPA's evaluation of our agent, its knowledge base was increased with 147 concepts, 104 tasks and 206 task reduction rules. This agent had the best correctness among all the teams that participated in the workaround challenge problem, and was selected to represent the HPKB program at EFX'98, the Air Force's show case of the most promising technologies.

One challenge problem for the second part of the HPKB program was to build a critiquing agent that can evaluate military Courses of Action (COA) that were developed as hasty candidate plans for ground combat operations. To solve this challenge problem we developed the Disciple-COA learning agent, and in the process we achieved two significant milestones with the Disciple approach. 1) For the first time we developed the knowledge base around an ontology created by another group (Teknowledge and Cycorp), demonstrating both the feasibility of knowledge reuse with the Disciple approach, and the generality of the Disciple rule learning and refinement methods. Moreover, the Disciple-COA agent was taught even more rapidly than the Disciple-workaround agent, and has demonstrated a higher performance than the other developed critiquers. 2) For the first time we conducted a knowledge acquisition experiment where four subject matter experts with no prior knowledge engineering experience received very limited training in the teaching of Disciple-COA and then each succeeded to significantly extend its KB, receiving only very limited support from a knowledge engineer (Tecuci et al., 2000).

5. Conclusions

In this paper we have briefly presented the Disciple approach, which represents a preliminary answer to the research question addressed in this paper.

With respect to this approach to agent development we formulate the following claims that have been tested during the intensive evaluations of the DARPA's HPKB program:

- it significantly speeds up the process of building and updating a high performance knowledge base;
- it enables rapid learning of problem solving knowledge from domain experts, with limited assistance from knowledge engineers;
- the learned problem solving knowledge is of a good enough quality to assure a high degree of correctness of the solutions generated by the agent;
- the acquired problem solving knowledge assures a high performance of the problem solver.

Our long-term vision is to develop a capability that will allow typical computer users to build and maintain knowledge bases and knowledge-based agents, as easily as they use personal computers for text processing or email.Therefore, this research aims at changing the way future knowledge-based agents will be built, from being programmed, to being taught.

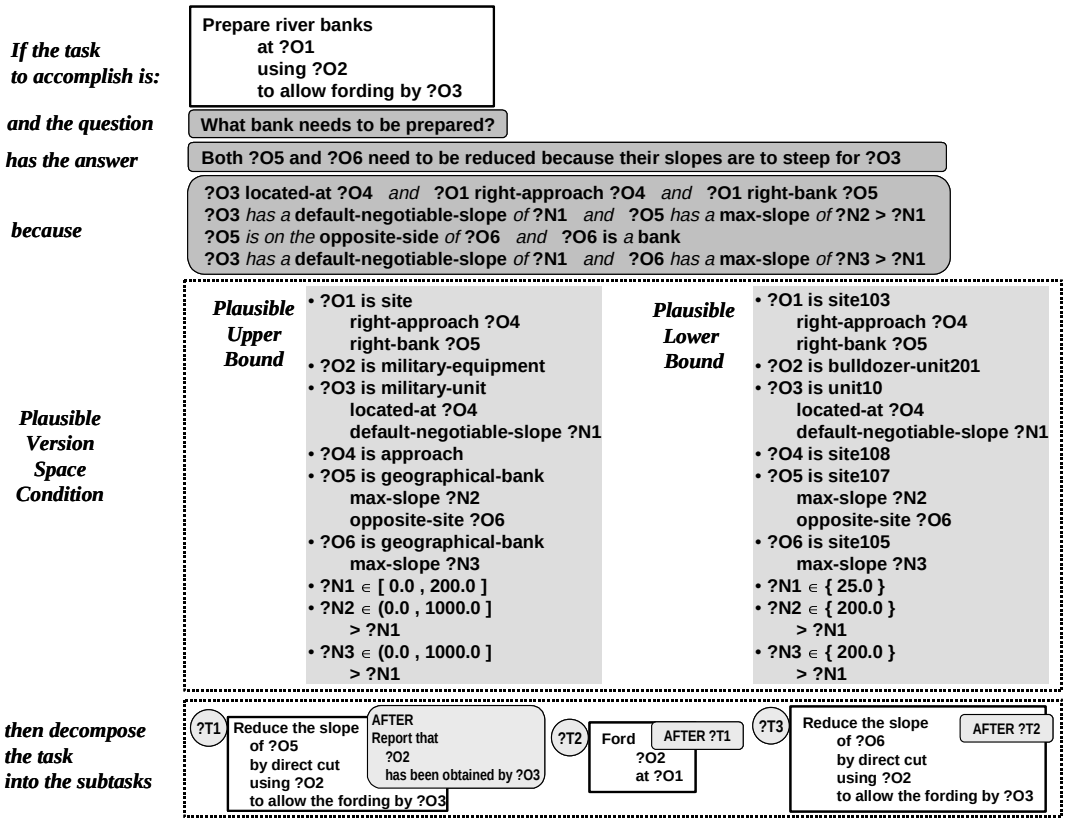

Figure 10: Partially learned task reduction rule

Acknowledgments.

This research was done in the GMU Learning Agents Laboratory (LALAB). Research of the LALAB is sponsored by the Defense Advanced Research Projects Agency (DARPA), the Air Force Office of Scientific Research (AFOSR) and Air Force Research Laboratory, Air Force Material Command, USAF, under agreement number F30602-00-2-0546, grant number F49620-97-1-0188 and grant number F49620-00-1-0072. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, AFOSR, the Air Force Research Laboratory, or the U.S. Government. Cascaval, C., Ciucu, F., Hamburger, H., Jenkins, T., Levcovici, C., Shyr, P., and Stanescu, B., have contributed to the current version of the Disciple approach.

References

Boicu, M., Tecuci, G., Bowman, M., Marcu, D., Lee, S. W. and Wright, K. 1999. A Problem-Oriented Approach to Ontology Development. In *Proceedings of the AAAI Workshop on Ontology Management*, 7-16, Menlo Park, California: AAAI Press.

Boicu, M., Tecuci, G., Marcu, D., Bowman, M., Shyr, P., Ciucu, F., and Levcovici, C. 2000. Disciple-COA: From Agent Programming to Agent Teaching. In *Proceedings of the Seventeenth International Conference of Machine Learning*, 73-80, San Francisco, California: Morgan Kaufmann Publishers.

Cohen, P., Schrag, R., Jones, E., Pease, A., Lin, A., Starr, B., Gunning, D., and Burke, M. 1998. The DARPA High-Performance Knowledge Bases Project. *AI Magazine* 19(4): 25-49.

Chaudhri, V. K., Farquhar, A., Fikes, R., Park, P. D., and Rice, J. P. 1998. OKBC: A Programmatic Foundation for Knowledge Base Interoperability. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 600–607, Menlo Park, California: AAAI Press.

Jones, E., 1998. HPKB Workaround Challenge Problem Specification, Alphatech, Inc., Burlington, MA.

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

Tecuci, G., Boicu, M., Wright, K., Lee, S. W., Marcu, D. and Bowman, M. 1999. An Integrated Shell and Methodology for Rapid Development of Knowledge-Based Agents. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 250-257, Menlo Park, California: AAAI Press.

Tecuci G., Boicu M., Bowman M., Marcu D., Shyr P., and Cascaval C. 2000 "An Experiment in Agent Teaching by Subject Matter Experts," *International Journal of Human-Computer Studies*, to appear.