

An Experiment in Agent Teaching by Subject Matter Experts

Gheorghe Tecuci, Mihai Boicu, Michael Bowman, Dorin Marcu, Ping Shyr, Cristina Cascaval

Learning Agents Laboratory, George Mason University, MSN 4A5,

4400 University Drive, Fairfax, VA 22030, USA

{tecuci, mboicu, mbowman3, dmarcu, ccascava}@gmu.edu, ShyrP@nasd.com

Abstract

This paper presents a successful knowledge acquisition experiment in which subject matter experts that did not have any prior knowledge engineering experience succeeded to teach the Disciple-COA agent how to critique courses of action, a challenge problem addressed by the DARPA's High Performance Knowledge Bases program. We first present the COA challenge problem and the architecture of Disciple-COA, a learning agent shell from the Disciple family. Then we present the knowledge acquisition experiment, detailing both the expert-Disciple interactions, and the automatic knowledge base development processes that take place as a result of these interactions. The results of this experiment provide strong evidence that the Disciple approach is a viable solution to the knowledge acquisition bottleneck.

1 Introduction

The development of large, easily maintainable and reusable knowledge bases for knowledge-based systems is one of the biggest challenges of Artificial Intelligence, with profound implications to its future. This problem has been a central one in two areas of artificial intelligence, Knowledge Acquisition and Machine Learning (Buchanan and Wilkins, 1993). However, each area has taken a different approach to this problem, as illustrated in Figure 1.

Knowledge Acquisition has taken the practical approach of building knowledge-based systems for real-world applications and has focused on improving and partially automating the acquisition of knowledge from subject matter experts by knowledge engineers. The knowledge engineer attempts to understand how the subject matter expert reasons and solves problems and then encodes the acquired expertise into the system's knowledge base (see Figure 1a). The expert analyzes the solutions generated by the system to identify errors and the knowledge engineer corrects the knowledge base (Gaines and Boose, 1988). This indirect communication of knowledge, from the domain expert through the knowledge engineer to the system (known as "the knowledge acquisition bottleneck") leads to a long, painful and inefficient system development process.

Machine Learning has taken a more theoretical approach, focusing on developing autonomous algorithms for acquiring knowledge from data and for knowledge compilation and organization (see Figure 1b). The difficulty of this task and the capabilities of the current autonomous learning methods limit the application of this approach to building less complex systems (Mitchell, 1997).

As Knowledge Acquisition attempts to automate more of the knowledge base development process and Machine Learning attempts to automatically develop more complex knowledge bases there is an increasing realization in these two research communities that a better approach to building knowledge bases is through an integration of different knowledge acquisition and learning methods, one that would take advantage of their complementary strengths to compensate for each other's weaknesses. As shown in Figure 1c, this approach relies on an integrated

Machine Learning and Knowledge Acquisition system. On the one hand, this system uses machine learning techniques to automate the knowledge acquisition from a team of subject matter experts and knowledge engineers. On the other hand, it uses knowledge acquisition techniques to enhance the power of the learning methods (Tecuci and Kodratoff, 1995).

The Machine Learning and Knowledge Acquisition system in Figure 1c performs several of the functions that, in the traditional Knowledge Acquisition approach (see Figure 1a), are performed by the knowledge engineer. Therefore, an even more challenging problem is the development of a knowledge base directly by a subject matter expert that has limited knowledge engineering experience and receives limited support from a knowledge engineer. This problem is addressed by the family of Disciple systems (Tecuci, 1998), a family of increasingly complex integrated Machine Learning and Knowledge Acquisition systems.

In this paper we present a knowledge acquisition experiment performed with the latest version of Disciple, called Disciple-COA. In this experiment subject matter experts succeeded in teaching Disciple-COA how to critique a military Course of Action (COA), while receiving very limited assistance from knowledge engineers. As a result of learning from a subject matter expert, Disciple-COA significantly extended its knowledge base. This experiment took place over one week, in August 1999, at the Battle Command Battle Lab, in Fort Leavenworth, Kansas, and was

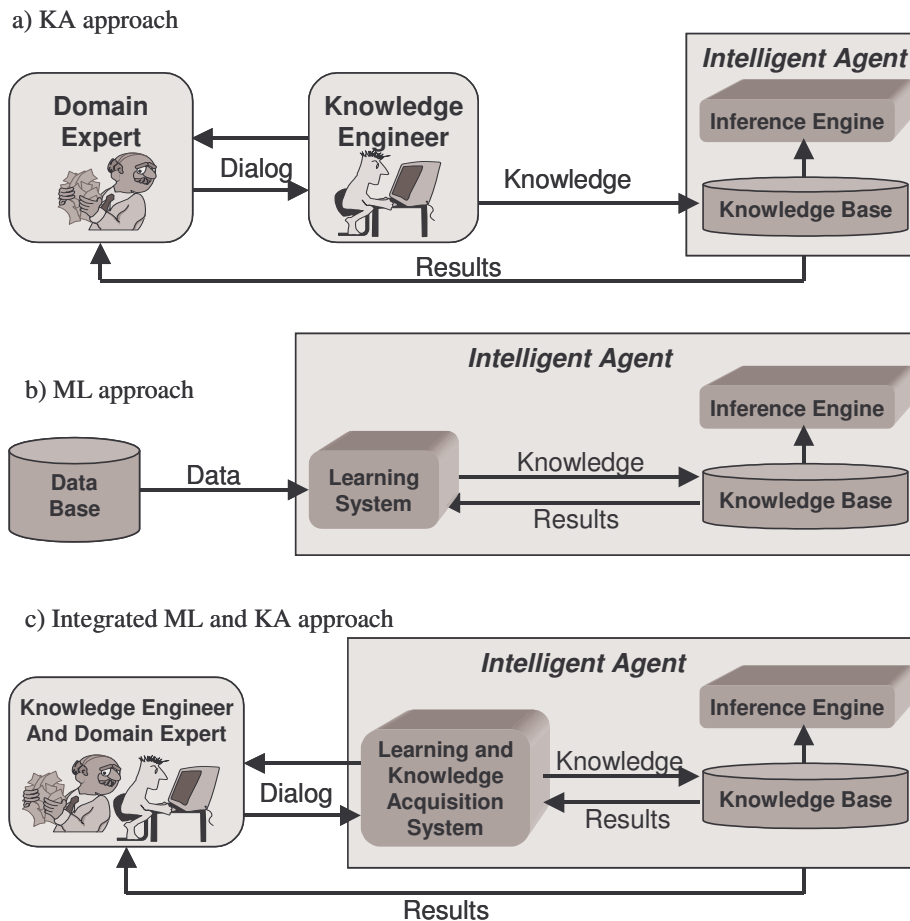


Figure 1: Approaches to the development of knowledge bases and knowledge-based agents.

part of DARPA's and AFOSR's High Performance Knowledge Bases (HPKB) program (Cohen, Schrag, Jones, Pease, Lin, Starr, Gunning and Burke, 1998).

The next section introduces the COA challenge problem for which the knowledge base of Disciple-COA was developed. Section 3 introduces Disciple-COA. Then section 4 gives an outline of the knowledge acquisition experiment. Sections 5 and 6 describe the knowledge acquisition experiment together with the knowledge representation, problem solving, learning and knowledge acquisition methods of Disciple-COA. After that, section 7 presents the results of the experiment and section 8 presents the conclusions of this paper.

2 The Course Of Action Challenge Problem

The COA challenge problem has been defined in the HPKB program to stimulate the development of advanced techniques for rapid development of large, easily maintainable and reusable knowledge bases for complex applications. A military COA is a preliminary outline of a plan for how a military unit might attempt to accomplish a mission. A COA is not a complete plan in that it leaves out many details of the operation such as exact initial locations of friendly and enemy forces. After receiving orders to plan for a mission a commander and staff complete a detailed and practiced process of analyzing the mission, conceiving and evaluating potential COAs, selecting of a COA, and then preparing detailed plans to accomplish the mission based on the selected COA. The general practice is for the staff to generate several COAs for a mission, and to then make a comparison of those COAs based on many factors including the situation, the commander's guidance, the principles of war, and the tenets of army operations. The commander makes the final decision on which COA will be used to generate his or her plan based on the recommendations of the staff and his or her own experience with the same factors considered by the staff (Jones, 1999).

The COA challenge problem consisted of rapidly developing a knowledge-based critiquing agent that can automatically critique COAs for ground force operations, systematically assess selected aspects of a COA, and suggest repairs to the COA. The input to the COA critiquing agent consists of the description of a COA that includes the following aspects:

- a) The COA sketch, such as the one in Figure 2, is a graphical depiction of the preliminary plan being considered. It includes enough of the high level structure and maneuver aspects of the plan to show how the actions of each unit fit together to accomplish the overall purpose, while omitting much of the execution detail that will be included in the eventual operational plan. The three primary elements included in a COA sketch are: control measures which limit and control interactions between units; unit graphics that depict known, initial locations and make up of friendly and enemy units; and mission graphics that depict actions and tasks assigned to friendly units. The COA sketch is drawn using a palette-based sketching utility.
- b) The COA statement, such as the partial one shown in Figure 3, clearly explains what the units in a course of action will do to accomplish the assigned mission. The text of a COA statement includes a description of the mission and desired end state, as well as standard elements that describe purposes, operations, tasks, forms of maneuvers, units, and resources to be used in the COA. The COA statement is expressed in a restricted but expressive subset of English.
- c) Selected products of mission analysis, such as the areas of operations of the units, avenues of approach, key terrain, units combat power, and enemy COAs.

Based on this input the critiquing agent has to assess various aspects of the COA, such as its viability (suitability, feasibility, acceptability and completeness), its correctness (array of forces, scheme of maneuver, command and control), and its strengths and weaknesses with respect to the principles of war and the tenets of army operations, to explain how it performed the assessments and to propose improvements to the COA.

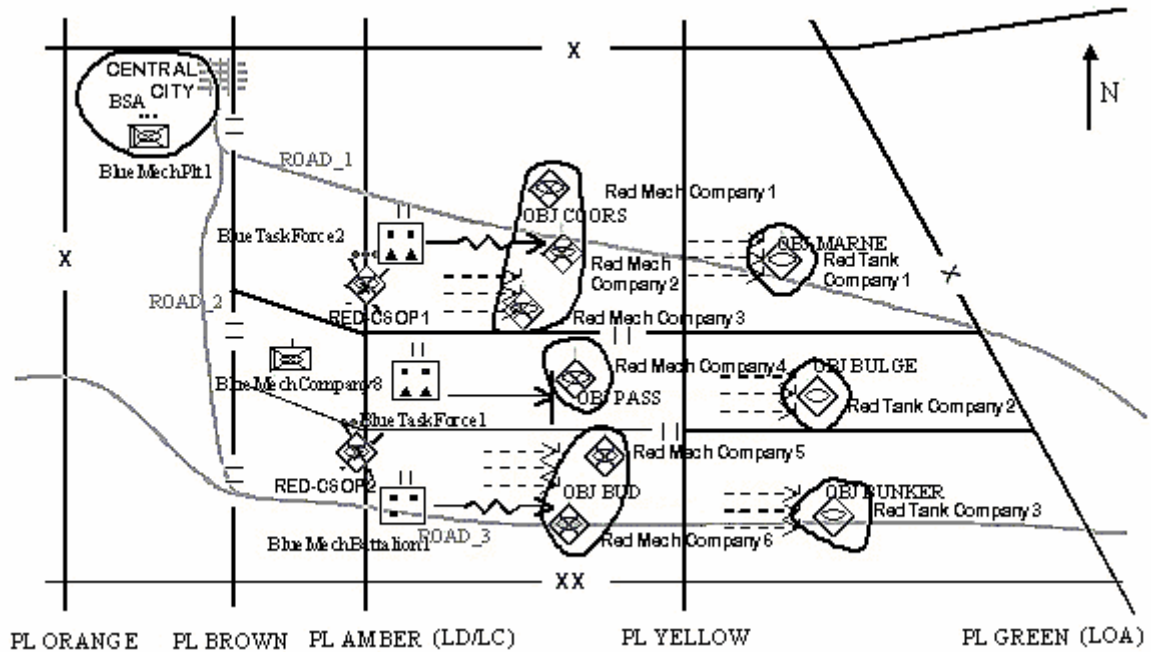


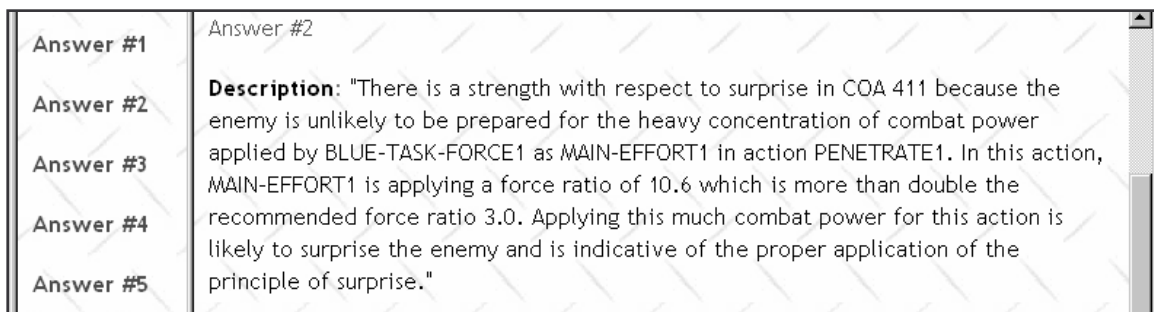
Figure 2: An example of a COA sketch.

<p>Mission: BLUE-BRIGADE2 attacks to penetrate RED-MECH-REGIMENT2 at 130600 Aug in order to enable the completion of seize OBJ-SLAM by BLUE-ARMOR-BRIGADE1.</p>
<p>Close: BLUE-TASK-FORCE1, a balanced task force (MAIN EFFORT1) attacks to penetrate RED-MECH-COMPANY4, then clears RED-TANK-COMPANY2 in order to enable the completion of seize OBJ-SLAM by BLUE-ARMOR-BRIGADE1.</p> <p>BLUE-TASK-FORCE2, a balanced task force (SUPPORTING EFFORT1) attacks to fix RED-MECH-COMPANY1 and RED-MECH-COMPANY2 and RED-MECH-COMPANY3 in order to prevent RED-MECH-COMPANY1 and RED-MECH-COMPANY2 and RED-MECH-COMPANY3 from interfering with conducts of the MAIN-EFFORT1, then clears RED-MECH-COMPANY1 and RED-MECH-COMPANY2 and RED-MECH-COMPANY3 and RED-TANK-COMPANY1.</p> <p>BLUE-MECH-BATTALION1, a mechanized infantry battalion (SUPPORTING EFFORT2) attacks to fix RED-MECH-COMPANY5 and RED-MECH-COMPANY6 in order to prevent RED-MECH-COMPANY5 and RED-MECH-COMPANY6 from interfering with conducts of the MAIN-EFFORT1, then clears RED-MECH-COMPANY5 and RED-MECH-COMPANY6 and RED-TANK-COMPANY3.</p> <p>... (descriptions of the other aspects of the COA)</p>

Figure 3: The Mission and Close sections of a COA statement.

In the HPKB program the COA challenge problem was addressed by building an integrated system composed of four critiquers developed by Teknowledge-Cycorp, ISI/Expect, ISI/Loom, and GMU/Disciple. All the critiquers shared an input and output ontology, and used the same input generated automatically by Teknowledge, AIAI and Northwestern Univ., from COA descriptions provided by Alphatech.

We have developed a COA critiquer, called Disciple-COA, which identifies the strengths and the weaknesses of a course of action with respect to the principles of war and the tenets of army operations (FM-105, 1993). There are nine principles of war: objective, offensive, mass, economy of force, maneuver, unity of command, security, surprise, and simplicity. They provide general guidance for the conduct of war at the strategic, operational and tactical levels. The tenets of army operations describe the characteristics of successful operations. They are: initiative, agility, depth, synchronization and versatility. Figure 4, for instance, shows a strength with respect to the principle of surprise, identified by Disciple-COA in the COA represented in Figures 2 and 3.



Answer #1	Answer #2
Answer #2	Description: "There is a strength with respect to surprise in COA 411 because the enemy is unlikely to be prepared for the heavy concentration of combat power applied by BLUE-TASK-FORCE1 as MAIN-EFFORT1 in action PENETRATE1. In this action, MAIN-EFFORT1 is applying a force ratio of 10.6 which is more than double the recommended force ratio 3.0. Applying this much combat power for this action is likely to surprise the enemy and is indicative of the proper application of the principle of surprise."
Answer #3	
Answer #4	
Answer #5	

Figure 4: A strength identified by Disciple-COA in the COA represented in Figures 2 and 3.

3 General presentation of Disciple-COA

Disciple is the name of an evolving theory, methodology and shell for rapid development of knowledge bases and knowledge-based agents, by subject matter experts, with limited assistance from knowledge engineers (Tecuci, 1998). The Disciple learning agent shell consists of a learning and knowledge acquisition engine as well as an inference engine and supports building an agent with a knowledge base consisting of an ontology and a set of problem solving rules. The Disciple shell was developed and customized into Disciple-COA, in order to address the COA challenge problem. The general architecture of Disciple-COA is presented in Figure 5.

The problem solving approach of Disciple is based on the task reduction paradigm. In this paradigm, a task to be accomplished by the agent (in this case the task of assessing a strength or a weakness of a COA) is successively reduced to simpler tasks until the initial task is reduced to a set of elementary tasks that can be immediately performed (i.e. assessments that can be immediately made). The Disciple-COA agent contains an autonomous COA critiquer that can generate all the strengths and weaknesses of the input COA, as shown in the bottom right of Figure 5. In addition, it also contains a cooperative, step-by-step, critiquer that allows Disciple to cooperate with a subject matter expert in critiquing a COA and to learn from the expert. The learning modules include a rule learner and a rule refiner, that are used to learn general task

reduction rules from specific reductions performed by the expert, and from their associated explanations.

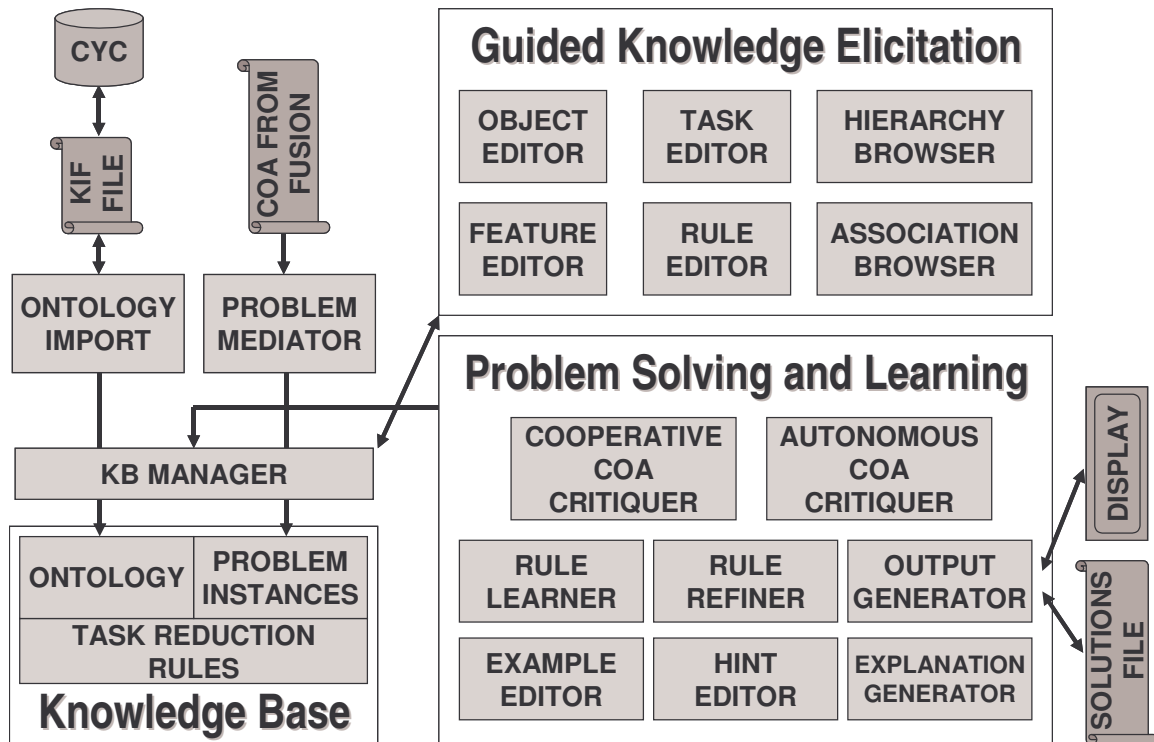


Figure 5: The architecture of Disciple-COA.

As shown in the bottom left corner of Figure 5, the knowledge base of Disciple-COA contains, in addition to a set of task reduction rules, an ontology and the problem instances. The ontology describes the general concepts from the COA critiquing domain (such as different types of military units, military tasks, operations and maneuvers, and geographical concepts). For Disciple-COA, an initial ontology was defined by importing relevant concepts from CYC (Lenat, 1995), using KIF as an intermediate language (Genesereth and Fikes, 1992). The ontology was further developed by using the ontology building tools of Disciple shown in the top right side of Figure 5 (the object, feature, task and rule editors and browsers). The problem instances describe the elements of the current COA to be critiqued. A file in the CYC language containing these descriptions is automatically generated from the input description of the COA (i.e. the text, the sketch and the products of mission analysis). This file is then automatically translated by the problem mediator into the Disciple language, and the resulting instances are introduced into Disciple's knowledge base.

4 Outline of the Knowledge Acquisition Experiment

The knowledge base of Disciple-COA was developed by a subject matter expert and a knowledge engineer and was evaluated in July 1999, in an intensive study, as part of the DARPA's HPKB annual evaluation. In August 1999 we performed a knowledge acquisition experiment with Disciple-COA. The main purpose of the experiment was to demonstrate that it is possible for a

military expert to directly teach Disciple how to critique a COA with respect to several principles of war and tenets of army operations. We also wanted to create a useful professional development experience for the participating military experts, beyond the use of Disciple. The experiment took place at the Battle Command Battle Lab in Fort Leavenworth, Kansas, between August 23, 1999 and August 27, 1999. There were four military experts participating in the experiment, none of which had previous knowledge engineering experience. The experiment had three phases: a joint training phase (day 1 to 3), an individual knowledge acquisition experiment (day 4), and a joint discussion of the experiment (day 5). The entire experiment was videotaped.

4.1 Training for the Experiment (Day 1 to 3)

We began the experiment with a general presentation of the experimentation goals and the schedule of the experiment. Then, in order to capture the interest of the domain experts, we showed each of them a COA and we asked them to analyze and critique it in about 30 minutes. After each expert generated their critique of the COA they were shown the critiques produced by Disciple-COA. The results of all the critiques were discussed by the group. A general consensus was that Disciple's critique was good with surprisingly detailed comments, but that it was not complete. When the critiques of each of the experts was discussed, it was observed that there were both common and unique aspects to their critiques. The overall conclusion was that each expert, and Disciple, had contributed correct elements to a complete critique but no single expert or Disciple had generated a comprehensive critique on their own.

We continued with a general presentation of Artificial Intelligence and of its evolution, linking it to the current state of the art and the current experiment. While this was not necessary in order to prepare the subject matter experts for the experiment, we felt that the experts would professionally benefit much more from this experiment by having this exposure.

The actual training for the experiment included a detailed presentation of Disciple's knowledge representation, problem solving and learning methods and tools. We also discussed how to model the process of critiquing a COA with respect to the tenet of initiative and the principle of surprise, including in this modeling some new aspects identified by the subject matter experts. Finally, the subject matter experts were shown how to use the modeling as a guide to teach Disciple to critique COAs.

4.2 The Individual Knowledge Acquisition Experiments (Day 4)

For the knowledge acquisition experiment itself each subject matter expert received a copy of Disciple-COA and several COAs. Each expert worked independently to teach Disciple and was assisted by a knowledge engineer when he encountered any difficulty in operating Disciple-COA.

4.3 Conclusion of the Experiment (Day 5)

In the last day of the experiment we had a discussion with the subject matter experts about the current status of Disciple and about some of the future research directions, receiving many useful comments and suggestions. The subject matter experts also filled-in a detailed questionnaire that is discussed in section 7.

5 Set-up of the Knowledge Acquisition Experiment

At the beginning of the experiment, each subject matter expert received a copy of Disciple-COA with a partial knowledge base. This knowledge base was obtained by removing the tasks and the

rules from the complete knowledge base of Disciple-COA. That is, the knowledge base contained the complete ontology of objects, object features, and task features. In the following we will describe each of these elements.

5.1 Disciple's ontology

Disciple's ontology includes objects, features and tasks, all represented as frames, according to the knowledge model of the Open Knowledge Base Connectivity (OKBC) protocol (Chaudhri, Farquhar, Fikes, Park and Rice, 1998).

The objects represent either specific individuals or sets of individuals. The objects are hierarchically organized according to the generalization relation (subclass-of/superclass-of and instance-of/type-of). Figure 6, for instance, presents several fragments of the object ontology used to model the COA domain. The top left part of Figure 6 represents the top level of the object ontology that identifies the types of concepts represented in the ontology. They include GEOGRAPHICAL-REGION, MODERN-MILITARY-ORGANIZATION, MILITARY-EQUIPMENT and MILITARY-TASK. Each of these concepts is the top of a specialized hierarchy. For instance, the bottom left of Figure 6 shows a fragment of the hierarchy of the MODERN-MILITARY-ORGANIZATION. The leaves of this hierarchy are specific military units, corresponding to a specific COA to be critiqued by Disciple. Each concept and instance of the object hierarchy is described by specific features and values. For instance, the top right side of Figure 6 shows the description of the specific military unit called BLUE-TASK-FORCE1. BLUE-TASK-FORCE1 is described as being both an ARMORED-UNIT-MILITARY-SPECIALTY and a MECHANIZED-UNIT-MILITARY-SPECIALTY. The other features describe BLUE-TASK-FORCE1 as being a battalion, belonging to blue side, being designated as the main effort of the blue side, performing two tasks, PENETRATE1 and CLEAR1, having a regular strength, and having under its operational control four other units. The values of the features of BLUE-TASK-FORCE1 are themselves described in the same way. For instance, one of the tasks performed by BLUE-TASK-FORCE1, PENETRATE1, is represented as shown in the bottom right of Figure 6. PENETRATE1 is defined as being a penetration task, and therefore inherits all the features of the penetration tasks, in addition to the features that are directly associated with it.

The hierarchy of objects is used as a generalization hierarchy for learning by the Disciple agent. One way to generalize an expression is to replace an object with a more general one from such a hierarchy. For instance, PENETRATE-MILITARY-TASK from the bottom right side of Figure 6 can be generalized to COMPLEX-MILITARY-TASK, or to MILITARY-MANEUVER, or to MILITARY-ATTACK. The goal of the learning process is to select the right generalization.

The features used to describe the objects and the tasks are themselves represented in the feature hierarchy. The top part of Figure 7 shows a fragment of the object hierarchy where the features "IS-SURPRISE-ACTION-FOR", "IS-SECURITY-ACTION-FOR" and "IS-OFFENSIVE-ACTION-FOR" are all subclasses of "IS-TYPE-OF-ACTION-FOR". The bottom part of Figure 7 shows a fragment of the task feature hierarchy.

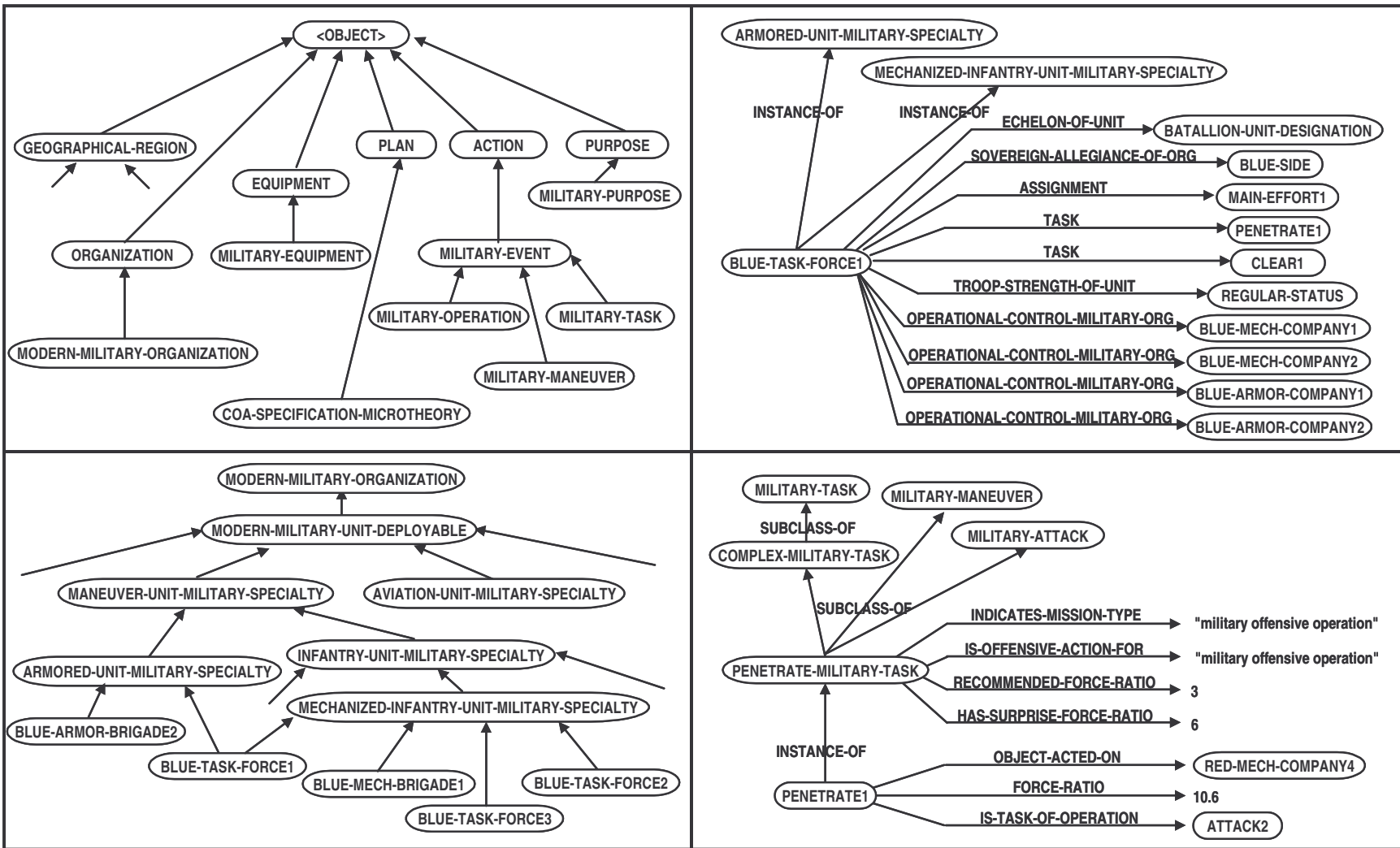


Figure 6: Fragments of the ontology imported from CYC.

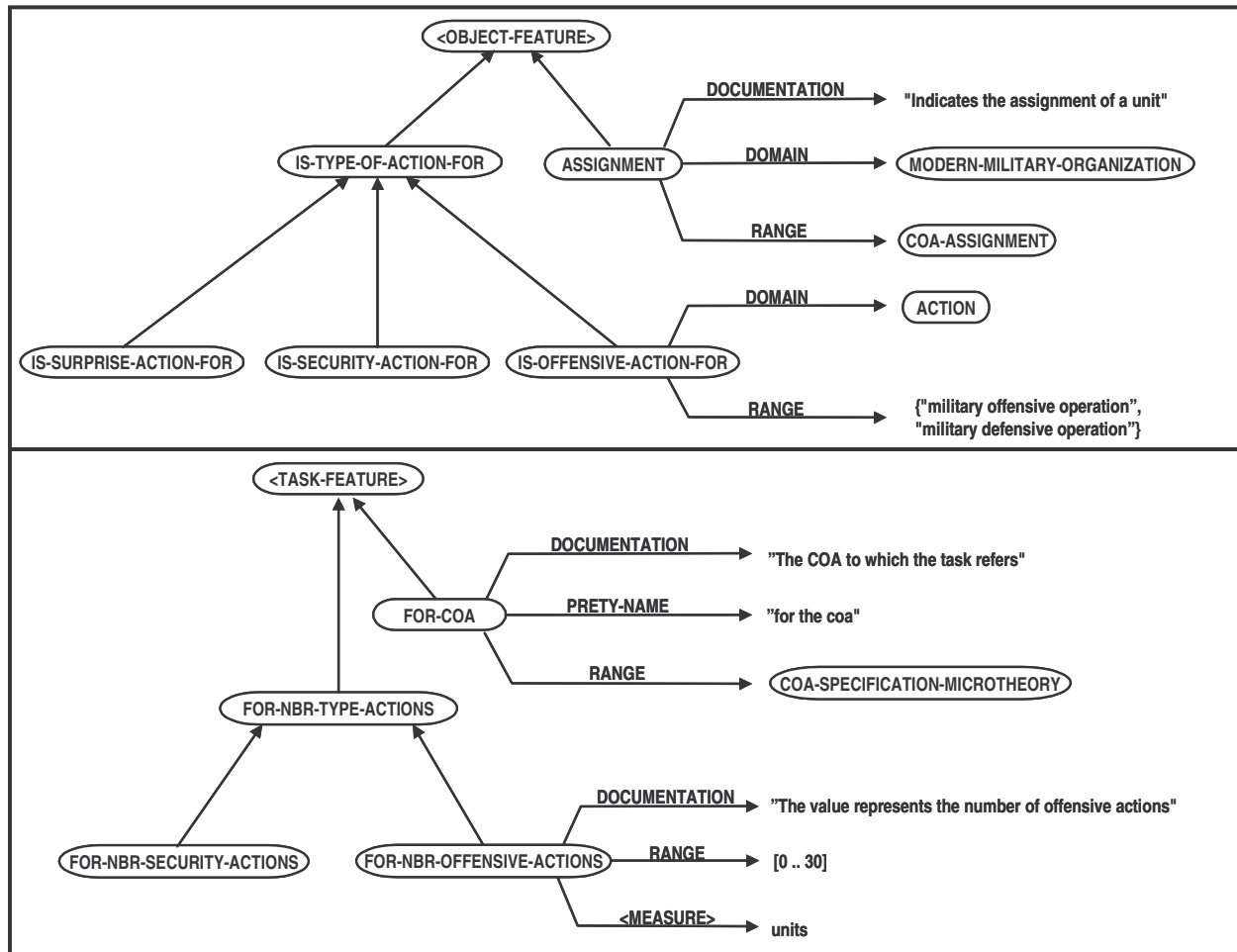


Figure 7: Sample hierarchies of object features (top) and task features (bottom).

Two important attributes of any feature are its domain (representing the set of objects that could have this feature) and its range (representing the set of possible values of the feature). For example, the DOMAIN of the feature ASSIGNMENT is MODERN-MILITARY-ORGANIZATION. This means that only subconcepts and instances of MODERN-MILITARY-ORGANIZATION may have the feature ASSIGNMENT. The RANGE of ASSIGNMENT is COA-ASSIGNMENT. Only subconcepts or instances of COA-ASSIGNMENT can be values of the feature ASSIGNMENT.

The features also play an important role in learning. Expressions can be generalized or specialized by adding or deleting features of the objects appearing in their descriptions. Also, the domains and the ranges of the features are guiding the generalizations of the expressions. For instance, the most general generalization that makes sense for the expression "BLUE-TASK-FORCE1 ASSIGNMENT MAIN-EFFORT" is "MODERN-MILITARY-ORGANIZATION ASSIGNMENT COA-ASSIGNMENT". That is, the most general generalization of BLUE-TASK-FORCE1 in the above expression is the domain of ASSIGNMENT. The feature generalization hierarchies are also used in analogical reasoning.

Many of the objects and object features from the Disciple COA ontology have been imported from the CYC's ontology (Lenat, 1995).

The knowledge base of Disciple-COA also contains tasks and rules that will be presented in more details in the following sections. A task is a representation of anything that the agent may be asked to accomplish, and is represented by a name and a set of task features. Each rectangle box in Figure 8 contains an example of a task. The tasks are also hierarchically organized, according to the more general than relation. For the knowledge acquisition experiment all the tasks were removed from the knowledge base, except for the top level ones that were predefined for the COA domain. However, the task features were kept to facilitate the definition of the tasks by the subject matter experts.

5.2 Modeling the COA assessment as task reduction

Before the knowledge acquisition experiment started, we discussed with the subject matter experts the modeling of the COA assessment as task reduction, for the two principles to be considered in the experiment, the Principle of Security and the Principle of Offensive. Figure 8, for instance, shows a fragment of this modeling for a specific COA (COA411 from Figures 2 and 3), and the Principle of Security. This modeling represents a sequence of reasoning steps through which a subject matter expert goes in order to assess COA411 with respect to the Principle of Security. The modeling is driven by a sequence of questions that the subject matter expert asks himself or herself and their corresponding answers. We start with the top-level task of assessing COA411 with respect to the Principle of Security. The first question has the purpose of enumerating the various aspects that one needs to consider with respect to the Principle of Security, and guides the subject matter expert to break the top level task into several subtasks, each corresponding to one of the identified aspects. One of these aspects is the enemy reconnaissance, and the corresponding subtask is to assess security of COA411 with respect to countering enemy reconnaissance. In order to perform this assessment the expert needs a certain amount of information about COA411, which is obtained by asking other questions. The first of these questions asks whether there is any enemy reconnaissance unit present in COA411. The answer identifies RED-CSOP1 as being such a unit because it is performing the task SCREEN1. Therefore, the task of assessing security of COA411 with respect to countering enemy reconnaissance is now reduced to the better-defined task of assessing security when the enemy reconnaissance is present.

The next question to ask is whether the enemy reconnaissance unit is destroyed or not. In the case of COA411, RED-CSOP1 is destroyed by the task DESTROY1. Therefore one can conclude that there is a strength in COA411 with respect to the Principle of Security because the enemy reconnaissance unit is destroyed. Each identified strength or weakness is assigned a certain importance by the subject matter expert, which could be low, medium or high. Following the other branches of the modeling in Figure 8 leads to the identification of additional strengths and weaknesses of COA411. An important aspect is that the above modeling corresponds to COA411, a specific COA, and therefore represents a specific problem solving case. However, because no single COA covers all the possible cases, the modeling is extended for additional COAs. All the questions and answers in this modeling are written in unrestricted natural language. The task names are also in natural language, but their features and values have to be from the knowledge base of Disciple.

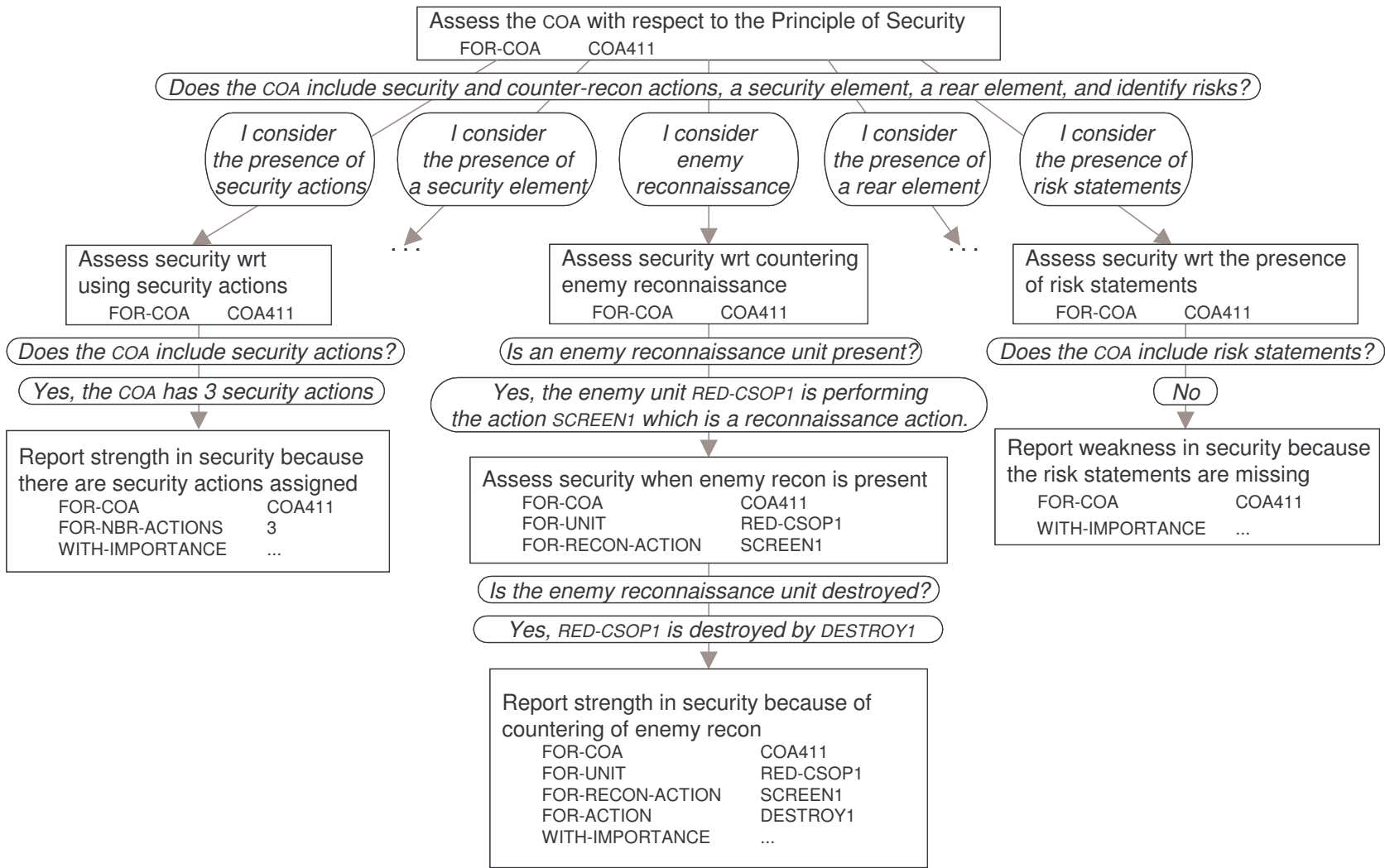


Figure 8: Modeling of the assessment of COA411 with respect to the Principle of Security.

6 The Knowledge Acquisition Experiment: Teaching the Disciple-COA agent

The modeling from Figure 8 guided each subject matter expert in teaching Disciple how to critique COAs with respect to the Principle of Security, as described in the following. The expert invoked the Cooperative Problem Solver, giving it the task from the top of Figure 8. The problem solver tried to reduce this task by applying the task reduction rules from its knowledge base. In principle, there are three possible outcomes:

A) No rule is applicable and therefore no reduction is proposed by Disciple. In this case the expert will provide a reduction, being guided by the domain modeling in Figure 8, and Disciple will learn a general reduction rule from expert's solution.

B) A reduction rule is applied to reduce the current task to a subtask, and the expert accepts this reduction. In this case the applied rule may be generalized, depending of how the reduction was generated.

C) A reduction rule is applied to reduce the current task to a subtask, but the expert rejects this reduction. In this case the applied rule is specialized to no longer generate the wrong reduction. Each of these cases is discussed in more detail in the following.

At the beginning of the knowledge acquisition experiment Disciple did not have any rule in its knowledge base. Therefore the reductions were given by the expert by following the modeling in Figure 8. Figure 9 shows the screen of the Cooperative Problem Solver at the moment when Disciple attempted to reduce the task "ASSESS-SECURITY-WRT-COUNTERING-ENEMY-RECONNAISSANCE." Because no reduction was proposed by Disciple, the only option for the expert was to click on the "New example" button. This invoked the Example Editor shown in Figure 10.

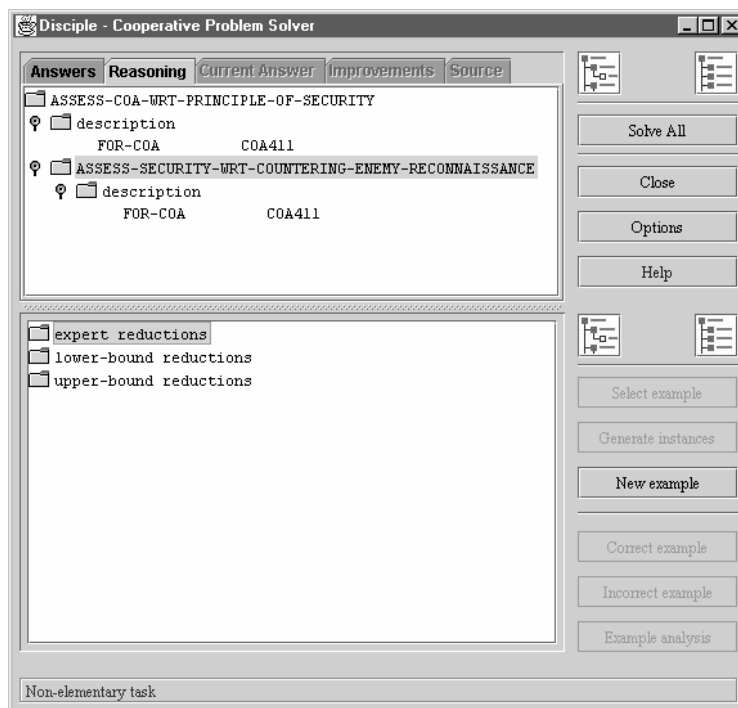


Figure 9: The interface of the Cooperative Problem Solver.

The top left pane in Figure 10 contains the task to be reduced. This is automatically filled-in by Disciple. The right panes are for the question and the answer corresponding to this task. The expert will need to fill them in from the provided modeling (see Figure 8). However, both the question and the answer are natural language expressions, and the expert may freely modify them. In the bottom left pane the expert has to provide the subtask, by following its description in the modeling. Again the expert can modify the name of the subtask. A result of this process is that a new task definition is added to Disciple's ontology:

```
ASSESS-SECURITY-WHEN-ENEMY-RECON-IS-PRESENT
  FOR-COA           ?O1
  FOR-UNIT          ?O2
  FOR-RECON-ACTION ?O3
```

Once the example reduction is completely defined in the Example Editor, the Rule Learner is invoked to learn a rule from the current example, as shown in Figure 11. Initially, the top left pane contains the example and the two right panes contain the question and the answer. Now the expert will have to help Disciple to understand why the task of assessing security of COA411 with respect to (wrt) countering enemy reconnaissance is reduced to one of assessing security when the enemy reconnaissance is present. This explanation is represented in the top left pane following the "BECAUSE" keyword. The explanation is, in fact, a formal representation of the information from the question and the answer. Indeed, the first explanation piece identifies RED-CSOP1 as an enemy unit:

```
RED-CSOP1 SOVEREIGN-ALLEGIANCE-OF-ORG RED--SIDE
```

The second explanation piece indicates that RED-CSOP1 performs the action SCREEN1:

```
RED-CSOP1 TASK SCREEN1
```

Finally, the last explanation piece states that SCREEN1 is a reconnaissance action:

```
SCREEN1 IS-AT-MOST INTELLIGENCE-COLLECTION-MILITARY-TASK
```

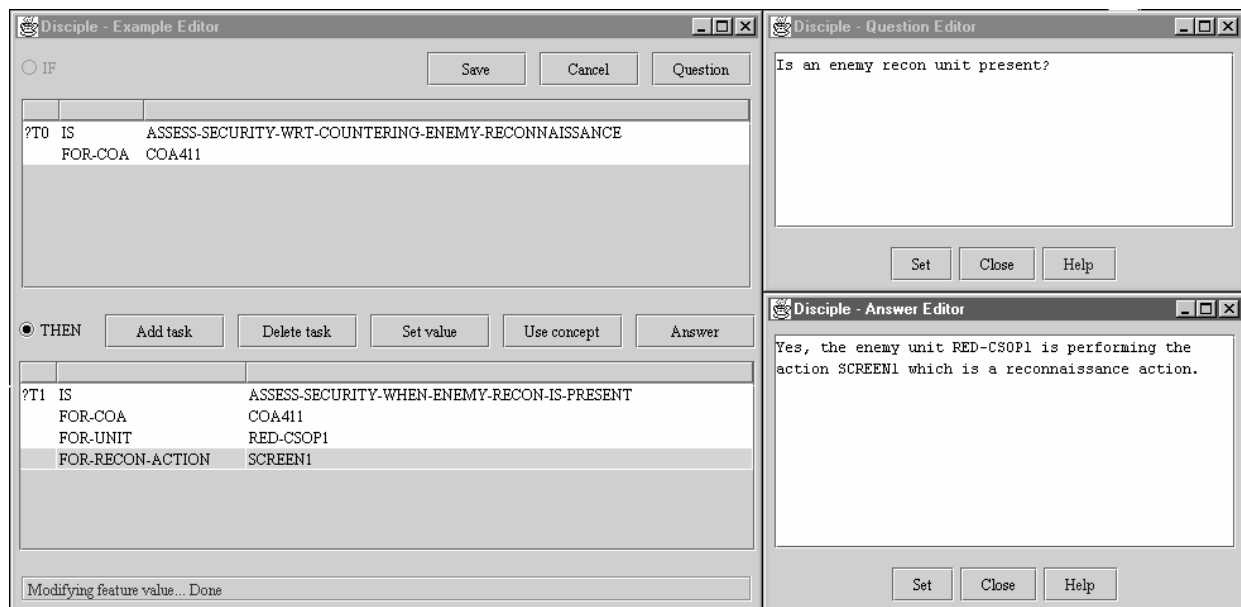


Figure 10: The interface of the Example Editor.

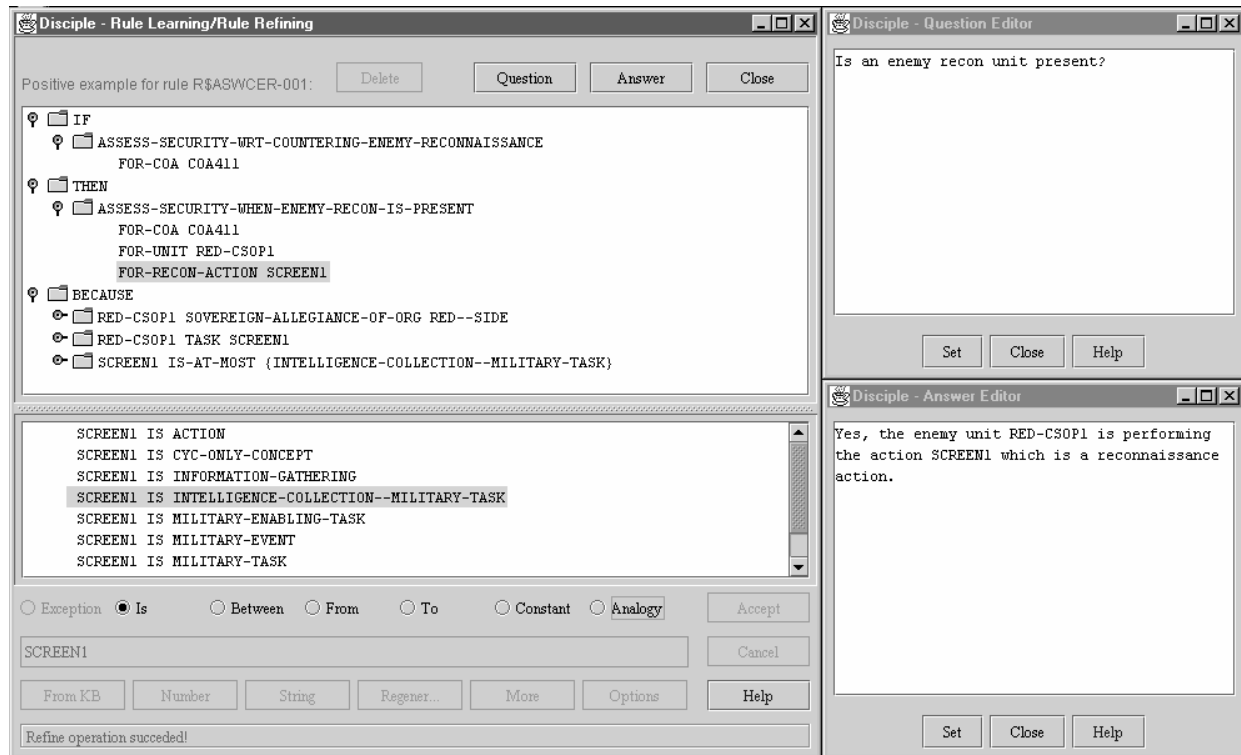


Figure 11: The interface of the Rule Learner.

However, Disciple-COA does not have the ability to understand natural language and to translate the question and the answer into the corresponding explanation pieces. Also, the expert cannot formulate the explanation by himself. First of all there are many hundreds of object and feature names in Disciple's knowledge base (such as "SOVEREIGN-ALLEGIANCE-OF-ORG") and the experts were not familiar enough with them. Secondly, the experts were not familiar enough with the formal syntax of Disciple, to be able to correctly define formal explanations. During their training, the experts were taught how to give Disciple hints that will help it to generate the explanations. Disciple can use these hints, explanation generation heuristics and analogical reasoning to propose a set of plausible explanations from which the expert will have to select the correct ones.

There are five main types of hints in the current version of Disciple: "Between", "From", "To", "Is", and "Constant". From the Answer, we know that it is important that "RED-CSOP1 is performing the action SCREEN1". We can simply select the "Between" button, and then point to "RED-CSOP1" and "SCREEN1" in the example. This directs Disciple to look for the paths between "RED-CSOP1" and "SCREEN1" in the object ontology. These paths represent plausible explanations and are displayed in the bottom left pane of the rule learning interface. One of these paths is "RED-CSOP1 TASK SCREEN1". The expert has to select it and, as a result, the explanation is moved in the top pane, under the BECAUSE keyword.

A "From" hint is given by selecting the "From" button and pointing to an object in the example, such as "RED-CSOP1". Disciple will look for an explanation in the form of a sequence of feature

values, starting with "RED-CSOP1". One of the generated explanations will be "RED-CSOP1 SOVEREIGN-ALLEGIANCE-OF-ORG RED-SIDE".

An "Is" hint indicates that a certain object should be of a certain type. The bottom left pane in Figure 11 shows the plausible explanations generated by Disciple as a result of receiving the hint that SCREEN1 should be of a certain type. Disciple shows the expert all the superclasses of SCREEN1 and the expert selects the right superclass: "SCREEN1 IS INTELLIGENCE-COLLECTION-MILITARY-TASK". The knowledge acquisition experiment demonstrated that the subject matter experts had no problems in giving hints to Disciple and in choosing the correct explanations from those generated by Disciple.

A powerful feature of Disciple-COA is its ability to generate plausible explanations of the current task reduction example using analogy with rules learned from similar examples. However, the use of this feature was more limited in the knowledge acquisition experiment because Disciple did not have many rules, all of them being learned during the experiment.

Based on the example reduction and its explanations shown in the top left pane of Figure 11, Disciple automatically generated the plausible version space rule shown in Figure 12. In essence, a rule is a complex IF-THEN structure that specifies one or several conditions under which the task from the IF part can be reduced to the task(s) from the THEN part. Each rule includes a main condition that has to be satisfied in order for the rule to be applicable. Partially learned rules, such as the ones showed in Figure 12, do not contain exact conditions but plausible version spaces for these conditions. Each such plausible version space is represented by a plausible upper bound condition which, as an approximation, is more general than the exact (but not yet known) condition, and a plausible lower bound condition which, as an approximation, is less general than the exact condition. Additionally, the rule may also include several "except-when" conditions (that should not hold in order for the rule to be applicable), "except-for" conditions (that specify instances that are negative exceptions of the rule) and "for" conditions (that specify positive exceptions). The generalizations of the explanations are also included in the learned rule. The rule also contains the generalizations of the natural language phrases representing the Question and its Answer from the example reduction. They are used by the natural language generation module of Disciple to generate the question and the answer part of a task reduction step obtained by instantiating a rule.

The IF-THEN rule is automatically created from the example by replacing each object with a different variable and restricting the possible values of these variables (Tecuci, 1998). The plausible lower bound condition, for instance, restricts the variables to only take values from the current example. It also includes the relations between the variables that have been identified as relevant in the explanation of the example. The plausible upper bound condition is the most general generalization of the plausible lower bound condition. It is obtained by taking into account the domains and the ranges of the features from the plausible lower bound conditions and the tasks in order to determine the possible values of the variables. For instance, ?O2 is the value of the task feature "FOR-UNIT", and has as features "SOVEREIGN-ALLEGENCE-OF-ORG" and "TASK". Therefore, any value of ?O2 has to be in the intersection of the range of "FOR-UNIT", the domain of "SOVEREIGN-ALLEGENCE-OF-ORG", and the domain of "TASK". This intersection is "MODERN-MILITARY-UNIT-DEPLOYABLE".

"expert reductions" Disciple will group reductions that have been previously provided by the expert for the current task to be reduced.

Many of the rules learned from critiquing COA411 were used by Disciple to critique another COA, COA421. As an illustration, the middle part of Figure 13 shows the task reduction steps for critiquing COA421 with respect to the Principle of Security when considering enemy reconnaissance. The first two reductions were proposed by Disciple based on two rules learned from COA411. Because both reductions were accepted by the expert they represent positive examples of the rules that generated them. If any of these examples has been generated based on the upper bound condition of a rule the lower bound condition of that rule is automatically generalized, as little as possible, to cover the new examples and to remain less general or as general as the corresponding plausible upper bound. The subject matter expert is not involved at all in this generalization process. As a result of such a process, the rule from Figure 12 was automatically generalized by Disciple-COA to the rule from the left hand side of Figure 13.

The only problem-solving step that was contributed by the expert in the case of the situation from Figure 13 is the last one. Because the enemy unit is not destroyed there is a weakness in the current COA. Disciple again attempted to find an explanation of this reduction. An explanation, however, is a piece of knowledge from the current ontology. The fact that RED-CSOP2 is not destroyed is represented by the absence of any action to destroy RED-CSOP2. This means that there is no knowledge piece in Disciple's ontology that would explicitly represent the fact that RED-CSOP2 is not destroyed. Therefore, Disciple cannot find this explanation. The only explanation provided is that the importance of such a discovered weakness should always be considered as high (an explanation of type "Constant"). As a consequence Disciple will learn an overly general rule from this example. This rule will be specialized when it will generate wrong solutions, as will be explained below.

In the knowledge acquisition experiment, after Disciple was taught to critique a new COA (such as COA421) the problem solver was invoked to apply the newly learned rules on the previously considered COAs (such as COA411) to provide a test case for these new rules. Figure 14 shows a screen of the cooperative problem solver for the case where COA411 was critiqued again, this time using also the rules learned from COA421. The expert noticed that this time Disciple generated two contradictory reductions of the task of assessing security when the enemy recon is present. In this case the expert selected the wrong reduction and clicked on the "Incorrect Example" button. This automatically invoked the Rule Refiner, as shown in Figure 15. Disciple used analogy in order to find the explanations of why the reduction is incorrect. In particular, it looked for explanations that were similar with the explanations of the other rules that reduce the task "ASSESS-SECURITY-WHEN-ENEMY-RECON-IS-PRESENT". It found two explanation pieces and displayed them in the bottom pane of the Rule Refiner interface. Then the expert selected the correct explanation: "DESTROY1 OBJECT-ACTED-ON RED-CSOP1".

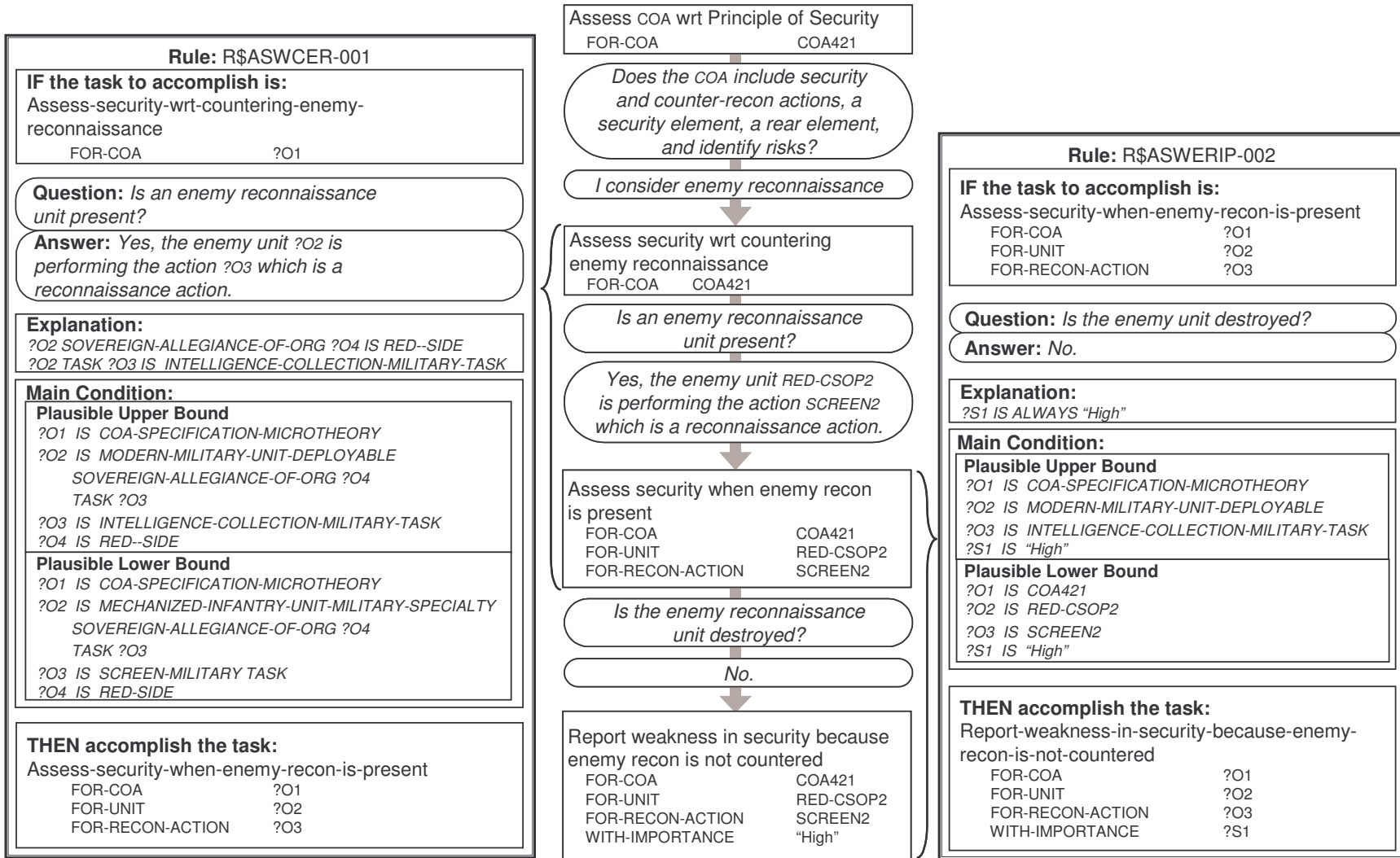


Figure 13: Illustration of cooperative problem solving and learning.

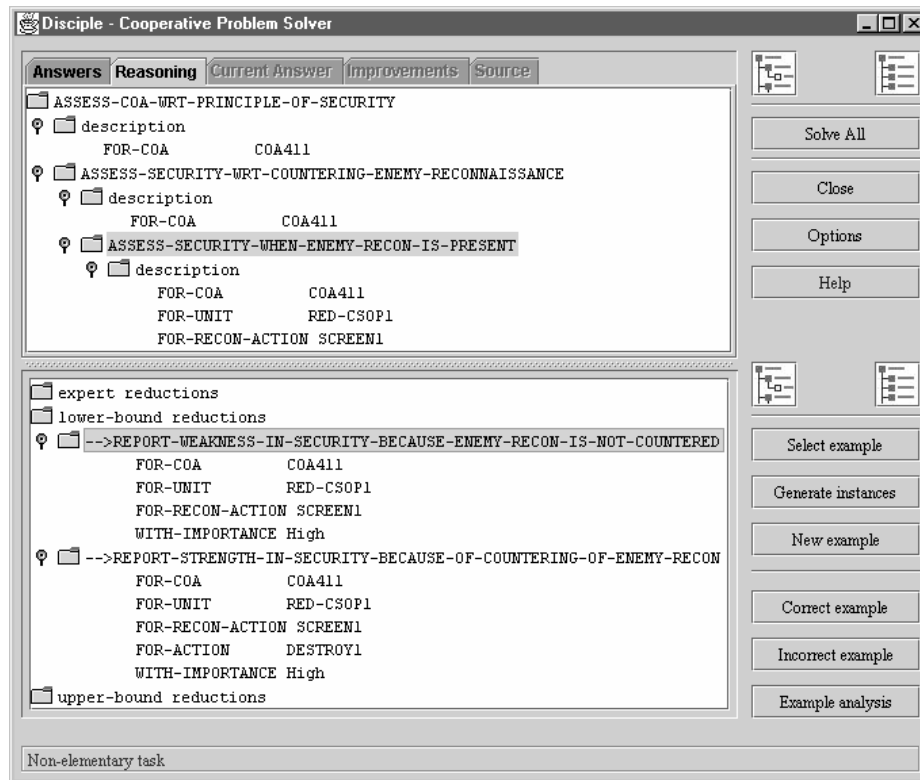


Figure 14: Cooperative Problem Solver Interface.

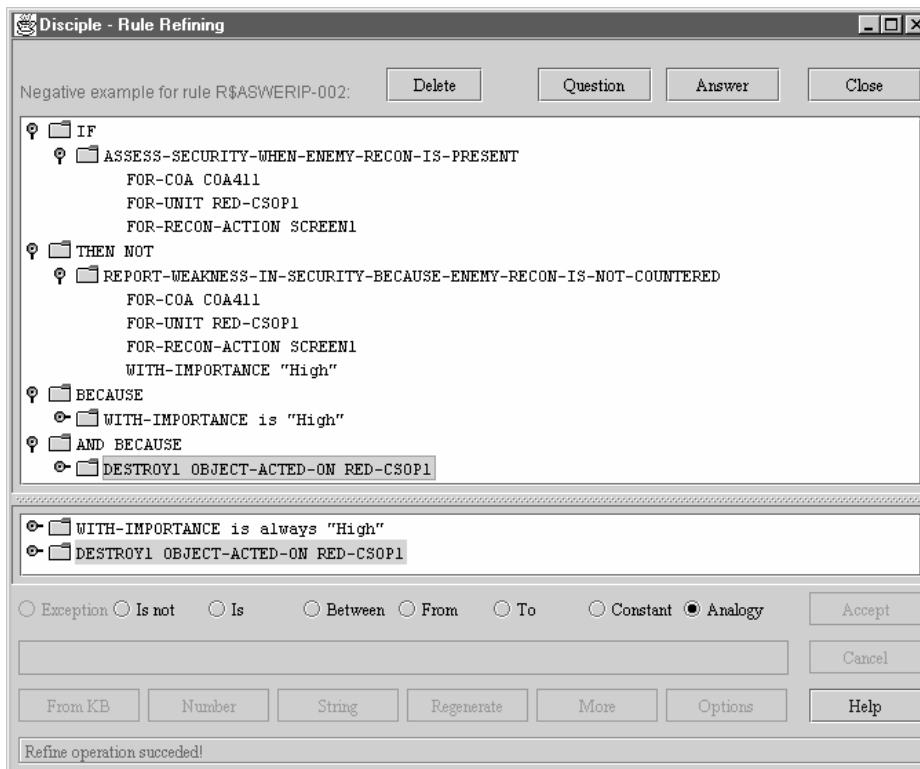


Figure 15: Rule Refiner Interface.

As indicated above, to define an example from which Disciple learns a rule the expert has to define the tasks that compose it. These tasks are added to the Disciple's ontology. Non-terminal tasks are characterized by a name, a set of features and, optionally, by a natural language description. The modeling in Figure 8 provides the expert with specific examples of such tasks and Disciple will automatically generalize them. For the terminal tasks the subject matter expert had to provide additional information that was not included in the received modeling. The subject matter expert has to provide the natural language description of the task, another natural language phrase representing a suggestion for improving the current COA (when the task was a weakness report), and a natural language phrase indicating the source material (such as specific references to military manuals) that support the answer represented by the terminal task. All these natural language phrases may refer to the objects from the task. Disciple generalized the specific task descriptions into general ones, as shown in Figure 17.

Task:	
REPORT-WEAKNESS-IN-SECURITY-BECAUSE-ENEMY-RECON-IS-NOT-COUNTERED	
FOR-COA	?O1
FOR-UNIT	?O2
FOR-RECON-ACTION	?O3
WITH-IMPORTANCE	?S1
NL description:	
There is a weakness with respect to security in COA <?O1> because there are no actions taken to destroy <?O2> which is an enemy unit assigned to collect intelligence and protect against surprise. The COA fails to call for aggressive security/counter-reconnaissance actions, destroying enemy intelligence collection units and activities. This and similar actions would enhance security by preventing the enemy from ascertaining the nature and intent of friendly operations, thereby decreasing the likelihood that the enemy will be prepared to counter friendly operations.	
NL improvement:	
In COA <?O1> plan for and conduct aggressive security/counter-reconnaissance actions, destroying enemy intelligence collection units and activities.	
NL source:	
FM 100-5 pg 2-5, KF 117.1, KF 117.3 - A strength with respect to security exists when action is taken that limits enemy opportunity to acquire unexpected advantage, thereby reducing friendly vulnerability to hostile acts, influence or surprise.	

Figure 17: Task description learned by Disciple

7 Experimental results

As mentioned in section 4, the subject matter experts were provided with 3 specific COAs to be used in teaching Disciple: COA411, COA421, and COA51. In the morning of the 4th day each expert used them to teach a personal copy of Disciple to critique COAs with respect to the Principle of Offensive and in the afternoon each expert used them to teach Disciple to critique COAs with respect to the Principle of Security.

From critiquing COA411 with respect to the Principle of Offensive, Disciple learned a number of rules from a subject matter expert. Then COA421 was loaded into the knowledge base of Disciple and the subject matter expert invoked the automatic problem solver to critique it by using the rules learned from COA411. Disciple correctly identified a strength in COA421. However, there

was one additional strength and one weakness in COA421. Therefore the expert invoked the Cooperative Problem Solver and taught Disciple to identify them, as presented in the previous section. Then COA51 was loaded into the system and the automatic problem solver was again used to critique it. In this case Disciple generated two wrong critiques. The subject matter expert used the Cooperative Problem Solver and analyzed each of the problem solving steps of the incorrect solutions, identifying the wrong task reductions and helping Disciple to understand why they were wrong. After that COA411 was again loaded into the system and critiqued with the updated rules. All the generated solutions were correct. The same was done for COA421, and again all the generated solutions were correct. The same procedure was repeated for the Principle of Security.

Figure 18 shows the evolution of the knowledge base during the teaching process for one of the experts, being representative for all the four experts. The horizontal line shows the successive COAs used to teach Disciple, and therefore corresponds to the time. As one can see from Figure 18, Disciple initially learned more tasks and rules, and then the emphasis shifted to rule refinement. Therefore, the increase in the knowledge base size is greater toward the beginning of the training process for each principle. The teaching for the Principle of Offensive took 101 minutes. During this time Disciple learned 14 tasks and 14 rules (147 simple axioms equivalent). The teaching for security consisted of 72 minutes of expert-Disciple interactions. During this time Disciple learned 14 tasks and 12 rules (136 simple axioms equivalent). There was very limited assistance from the knowledge engineer with respect to teaching. The knowledge acquisition rate obtained during the experiment was very high (~ 9 tasks and 8 rules / hour expert, or 98 simple axioms equivalent/hour). At the end of this training process, Disciple-COA was able to correctly identify 17 strengths and weakness of the 3 COAs with respect to the principles of offensive and security.

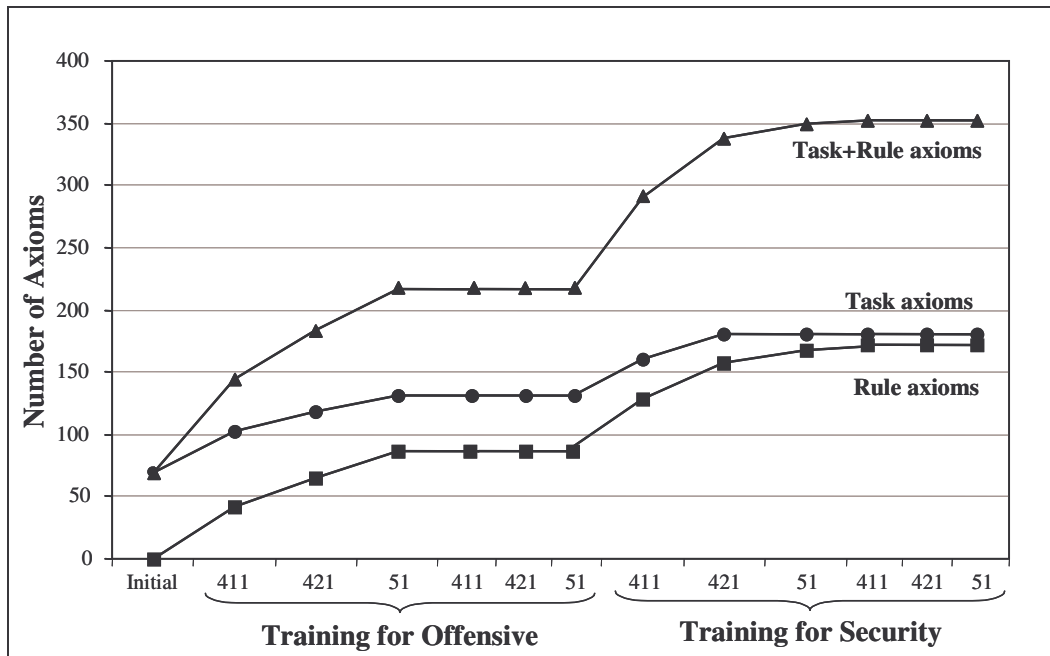


Figure 18: The evolution of the knowledge base during the knowledge acquisition experiments.

After the experiment, each expert was asked to fill in a detailed questionnaire designed to collect subjective data for usability evaluation. The questionnaire was organized into three major sections: a) 6 overall questions, b) 40 detail questions, and c) comments and recommendations. The questions addressed three main dimensions of evaluation: effect on task performance, system usability, and system fit. Each dimension considered various criteria (e.g. system fit with the user or system fit with the organization) and even sub-criteria. All answers took into account that Disciple-COA was a research prototype and not a commercial product. They were rated based on a scale of agreement with the question from 1 to 5, with 1 denoting no agreement at all and 5 denoting full agreement. For illustration, Figure 19 shows five of the most relevant questions and the answers provided by the four experts.

<i>Questions</i>	<i>Answers</i>
Do you think that Disciple is a useful tool for Knowledge Acquisition?	<ul style="list-style-type: none"> • Rating 5. Absolutely! The potential use of this tool by domain experts is only limited by their imagination - not their AI programming skills. • 5 • 4 • Yes, it allowed me to be consistent with logical thought.
Do you think that Disciple is a useful tool for Problem Solving?	<ul style="list-style-type: none"> • Rating 5. Yes. • 5 (absolutely) • 4 • Yes. As it develops and becomes tailored to the user, it will simplify the tedious tasks.
Do you think that Disciple has potential to be used in other tasks you do?	<ul style="list-style-type: none"> • Rating 5. Again the use of the tool is only limited to one's imagination but potential applications include knowledge bases built to support distance/individual learning, a multitude of decision support tools (not just COA Analysis), and autonomous and semi-autonomous decision makers - <u>all these designed by the domain expert vs an AI programmer.</u> • Absolutely. It can be used to critique any of the BOS's for any mission • 5 Yes • 4
Were the procedures/ processes used in Disciple compatible with Army doctrine and/or decision making processes?	<ul style="list-style-type: none"> • Rating 5. As a minimum yes, as a maximum—better! • This again was done very well. • 4 • 4
Could Disciple be used to support operational requirements in your organization?	<ul style="list-style-type: none"> • Rating 5. Yes, Absolutely! I'll take 10 of them! • 5 • 5 • Not at this point of development.

Figure 19: Sample questions and their answers.

8 Discussion and Conclusions

In this paper we have presented a knowledge acquisition experiment where subject matter experts that had no prior knowledge engineering experience extended the knowledge base of Disciple-COA, without receiving any significant assistance from knowledge engineers. The knowledge acquisition experiments are very expensive because they require a significant amount of time from highly trained experts. Therefore we had to limit the experiment to what was feasible to be done in one week. The experts received a knowledge base that contained all the necessary objects

and features. They also received the modeling for critiquing three COAs with respect to two principles of war, offensive and security. With this information they successfully extended the knowledge base of Disciple with tasks and task reduction rules, to critique COAs with respect to the two principles.

Why does the Disciple approach work? Comparing the simplicity of the task reductions from the middle of Figure 13, reductions defined by the subject matter experts or proposed by Disciple, with the complexity of the rules learned by Disciple from them, provides an answer to this question: Disciple reduces the complex operations that are necessary in order to build a knowledge base (operations that require a knowledge engineer and cannot be performed by a subject matter expert), to simpler operations that can be performed by a subject matter expert with limited or no assistance from a knowledge engineer. Indeed, to build a knowledge base, a knowledge engineer has to define an ontology, has to define problem solving rules, has to verify and update both the rules and the ontology.

In general, the knowledge engineer has to create formal sentences, including formal explanations of system's behavior. In the Disciple approach, all these operations are replaced with simpler ones. Rather than defining an ontology, one can import it from a repository of knowledge and update it for the current application. Rather than defining complex problem solving rules, one only needs to define examples because Disciple can generalize them into rules. Rather than verifying complex rules, one only needs to critique the examples of these rules because Disciple can update the rules accordingly. Rather than creating sentences in a formal language, one only needs to understand the sentences generated by Disciple. Rather than defining formal explanations, one only needs to provide hints that will guide Disciple in generating the explanations. The knowledge acquisition experiment demonstrated that some of these operations can indeed be performed by a subject matter expert. In particular, it showed that a subject matter expert can define examples of task reductions, can provide hints to the system, and can understand formal expressions (representing either examples or explanations). We have not addressed in this experiment the updating of the ontology. Also, the definition of examples was controlled by providing the modeling to the experts. However, during the training of the subject matter experts, they successfully extended the modeling of other principles and tenets.

The general philosophy of Disciple, namely to reduce complex knowledge engineering operations to simpler ones, is similar in spirit to that of AQUINAS and KSS0 (Gaines and Shaw, 1992). These systems elicit repertory grids from experts through a very natural dialog.

From the point of view of the learning methods and techniques employed, this research is mostly related to the work on learning apprentices. Learning apprentices are interactive knowledge-based consultants that are able to assimilate new knowledge by observing and analyzing the problem solving steps contributed by their expert users through their normal use of the agents. Examples of such learning agents are LEAP (Mitchell et al, 1990; Mahadevan et al. 1993), Odysseus (Wilkins, 1993), Clint (DeRaedt, 1991) and CAP (Mitchell et al, 1994). Disciple-COA is itself a learning apprentice. However, it is the only one to specifically address the issues involved in learning directly from a subject matter expert. The type of user was not a concern in the previous learning apprentices. As in many other cases of machine learning research the assumption was that the learner always receives the input it needs, with no real concern of how and by whom this input is provided. Disciple-COA also learns from its user in a way that is very different from the other developed apprentices. In particular, it synergistically integrates several

specific learning strategies (such as learning from examples, learning from explanations, learning by analogy, learning by experimentation) and uses more types of interactions with the expert than the other developed apprentices. The types of user-learner interactions are quite limited in the developed interactive learning systems, generally the user being regarded as an example initiator or as an oracle. Finally, Disciple-COA has been scaled up and developed into an integrated shell and methodology for building practical end-to-end agents. This process includes not only problem solving and learning of rules, but also domain modeling and ontology development.

In conclusion, the results of the knowledge acquisition experiment presented in this paper are very encouraging, providing strong evidence that the Disciple approach is a viable solution to the knowledge acquisition bottleneck. It also demonstrates the feasibility of our long-term vision for Disciple: to develop a capability that will allow normal computer users to build and maintain knowledge bases and knowledge based agents, as easily as they use personal computers for text processing or email.

Acknowledgments. This research was supported by AFOSR and DARPA through the grants F49620-97-1-0188 and F49620-00-1-0072. The evaluation of the COA critiquers in the HPKB program was conducted by Alphatech. The experts that participated in the BCBL knowledge acquisition experiment were LTC John N. Duquette, LTC Jay E. Farwell, MAJ Michael P. Bowman, and MAJ Dwayne E. Ptaschek. The organizers of the experiment were LTC Eugene F. Stockel and MAJ Robert A. Rasch, Jr. Gil Trenum provided useful comments on this paper.

References

- Boicu M., Wright K., Marcu D., Lee S.W., Bowman M. and Tecuci G. (1999). The Disciple Integrated Shell and Methodology for Rapid Development of Knowledge-Based Agents. In *AAAI-99/IAAI-99 Proceedings*, pp. 900–901, Menlo Park, CA: AAAI Press.
- Buchanan, B. G. and Wilkins, D. C. (editors), (1993). *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems*, Morgan Kaufmann, San Mateo, CA.
- Chaudhri, V.K., Farquhar, A., Fikes, R., Park, P.D., and Rice, J.P. (1998). OKBC: A Programmatic Foundation for Knowledge Base Interoperability. In *AAAI-98 Proceedings*, pp. 600–607, Menlo Park, CA: AAAI Press.
- 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
- De Raedt, L. (1991). *Interactive Concept Learning*. Ph.D. Thesis, Catholic University of Leuven.
- FM-105. (1993). US Army Field Manual 100-5, Operations, Headquarters, Department of the Army.
- Gaines, B.R. and Boose, J.H. Eds.(1988). *Knowledge Acquisition for Knowledge Based Systems*, Academic Press.

Gaines, B. R. and Shaw, M.L.G. (1992). Knowledge Acquisition Tools Based on Personal Construct Psychology, *Knowledge Engineering Review special issue on automated knowledge acquisition tools*.

Genesereth, M. R. and Fikes. R. (1992). Knowledge Interchange Format, Version 3.0 *Reference Manual*. Logic-92-1. Computer Science Department, Stanford University.

Jones, E., (1999). HPKB course of action challenge problem specification, Burlington, MA: Alphatech, Inc.

Kim, J. and Gil, Y. (1999). Deriving Expectations to Guide Knowledge Base Creation. In *AAAI-99/IAAI-99 Proceedings*, Menlo Park, CA: AAAI Press.

Lenat, D. B. (1995). CYC: A Large-scale Investment in Knowledge Infrastructure. *Communications of the ACM* 38(11), 33-38.

MacGregor, R. (1999). Retrospective on LOOM. Available online as: http://www.isi.edu/isd/LOOM/papers/macgregor/Loom_Retrospective.html.

Mahadevan, S., Mitchell, T., Mostow, J., Steinberg, L., and Tadepalli, P. (1993). An Apprentice Based Approach to Knowledge Acquisition, *Artificial Intelligence*, 64(1): 1-52.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Mitchell T.M., Mahadevan S. and Steinberg L.I. (1990). LEAP: A Learning Apprentice System for VLSI Design, in *Machine Learning: An Artificial Intelligence Approach*, Vol. 3, Y. Kodratoff and R.S. Michalski (Eds.), San Mateo, CA, Morgan Kaufmann.

Mitchell, T. M., Caruana, R., Freitag, D., McDermott, J. and Zabowski, D. (1994). Experience with a Learning Personal Assistant, *Communications of the ACM*, Vol. 37, pp. 81-91.

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

Tecuci, G. and Kodratoff, Y. (editors), (1995). *Machine Learning and Knowledge Acquisition: Integrated Approaches*, 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 *AAAI-99/IAAI-99 Proceedings*, Menlo Park, CA: AAAI Press.

Wilkins D.C. (1990). Knowledge Base Refinement as Improving an Incorrect and Incomplete Domain Theory, in Y. Kodratoff and R. S. Michalski (eds), *Machine Learning: An Artificial Intelligence Approach*, vol. 3, Morgan Kaufmann.