THE DISCIPLE-RKF LEARNING AND REASONING AGENT

Gheorghe Tecuci, Mihai Boicu, Cristina Boicu,

Dorin Marcu, Bogdan Stanescu, Marcel Barbulescu

MSN 4A5, Learning Agents Center and Computer Science Department,

George Mason University, 4400 University Drive, Fairfax, VA 22030, USA

{tecuci, mboicu, ccascava, dmarcu, bstanesc}@gmu.edu, mbarb@cs.gmu.edu

http://lac.gmu.edu, tel: 1 703 993-1722, fax: 1 703 993-1710

Abstract

Over the years we have developed the Disciple theory, methodology, and family of tools for building knowledge-based agents. This approach consists in developing an agent shell that can be taught directly by a subject matter expert, in a way that resembles how the expert would teach a human apprentice when solving problems in cooperation. This paper presents the most recent version of the Disciple approach and its implementation in the Disciple-RKF system. Disciple-RKF is based on methods for *mixed-initiative problem solving*, where the expert solves the more creative problems and the agent solves the more routine ones, *integrated teaching and learning*, where the agent helps the expert to teach it, by asking relevant questions, and the expert helps the agent to learn, by providing examples, hints and explanations, and *multistrategy learning*, where the agent integrates multiple learning strategies, such as learning from examples, learning from explanations, and learning by analogy, to learn from the expert how to solves problems. Disciple-RKF has been successfully applied to build learning and reasoning agents for military center of gravity analysis, which are used in several courses at the US Army War College.

Key Words: multistrategy apprenticeship learning, problem solving through task reduction, mixed-initiative reasoning, plausible version spaces, rule learning, ontology, agent development, military center of gravity analysis

1. INTRODUCTION

For almost 20 years we have performed research on developing a theory and the associated methodologies and tools for building agents that incorporate the knowledge of a subject matter expert (Tecuci 1988, 1998; Boicu 2002).

Our approach to this problem, which we have called the Disciple approach, consists in developing a problem solving and learning agent that can be taught directly by a subject matter expert to become a knowledge-based assistant. The expert should be able to teach the agent how to perform problem solving tasks in a way that is similar to how the expert would teach a person. For instance, the expert may show the agent how to solve specific problems, and may help it to understand the reasoning process. As the agent learns general problem solving rules from these problem solving examples and builds its knowledge base, the expert-agent interaction evolves from a teacher-student interaction toward an interaction where both collaborate in solving a problem. During this joint problem solving process, the agent learns not only from the contributions of the expert, but also from its own successful or unsuccessful problem solving attempts. This process is based on:

- mixed-initiative problem solving (Tecuci et al., 1993), where the expert and the agent solve problems in cooperation and the agent learns from the contributions of the expert;
- integrated learning and teaching, where the expert helps the agent to learn (for instance, by providing examples, hints and explanations), and the agent helps the expert to teach it (for instance, by asking relevant questions), and;
- multistrategy learning (Michalski and Tecuci, 1994), where the agent integrates complementary strategies, such as learning from examples, learning from explanations, and learning by analogy, to learn general concepts and rules.

Over the years we have continuously extended and improved the Disciple approach, which is reflected in a sequence of increasingly more powerful problem solving and learning agents from the Disciple family of agents. The goal of this paper is to overview the knowledge representation, problem solving and learning methods of the most recent member of this family, Disciple-RKF, which has been developed as part of the DARPA's Rapid Knowledge Formation (RKF) program (Tecuci et al., 2001).

The next section introduces the military center of gravity analysis domain which was used as the main application domain of Disciple-RKF and will provide examples for presenting Disciple-RKF. Section 3 presents the general architecture of Disciple-RKF, which integrates general components (usable across many application domains) and specific components (developed for a specific application domain to improve the usability and the efficiency of the agent). Section 4 presents the task reduction paradigm, the general problem solving approach of Disciple-RKF, which allows it to be used in a wide variety of domains (such as action planning, military course of action critiquing, or center of gravity analysis), and for several purposes (such as, modeling the expert's reasoning, mixed-initiative problem solving, and agent teaching and learning). Section 5 presents the knowledge representation of Disciple-RKF, which is based on the concept of plausible version space, allowing the agent to learn with an evolving representation language and to use partially learned knowledge in problem solving. This sets the stage for presenting the multistrategy learning methods of Disciple-RKF in section 6. These methods integrate learning from examples, learning from explanations, and learning by analogy. Section 7 presents an experiment with Disciple-RKF and its deployment at US Army War College. The paper concludes with a discussion of related research, current limitations of Disciple-RKF and, the main directions of future research.

2. SAMPLE APPLICATION DOMAIN:

MILITARY CENTER OF GRAVITY ANALYSIS

Military center of gravity analysis was used as a challenge problem in the DARPA's RKF program to test the knowledge acquisition, learning and problem solving methods of Disciple-RKF, and will be used in this paper to illustrate them. The concept of center of gravity, introduced by Karl von Clausewitz (1832), is fundamental to military strategy, denoting the primary source of moral or physical strength, power or resistance of a force (Strange, 1996). The most important objective of a force (state, alliance, coalition, or group), in any type of conflict, is to protect its own center of gravity while attacking the center of gravity of its enemy. Therefore, in the education of strategic leaders at all the U.S. senior military service colleges, there is great emphasis on the center of gravity analysis. This analysis requires a wide range of background knowledge not only from the military domain, but also from the political, psychosocial, economic, geographic, demographic, historic, international, and other domains (Giles and Galvin 1996). In addition, the situation, the adversaries involved, their goals, and their capabilities can vary in important ways from one scenario to another. Therefore, this is a very good example of knowledge-intensive, expert problem solving, that a Disciple agent should be able to learn.

Our approach to center of gravity analysis, based on the work of Strange (1996) and Giles and Galvin (1996), and developed with experts from the US Army War College, consists of two main phases: *identification* and *testing*. During the identification phase, center of gravity candidates from different elements of power of a force (such as government, military, people, economy) are identified. For instance, a strong leader is a center of gravity candidate with respect to the government of a force. Then, during the testing phase, each candidate is analyzed to determine whether it has all the critical capabilities that are necessary to be the center of

gravity. For example, a leader needs to be protected, stay informed, communicate (with the government, the military, and the people), be influential (with the government, the military, and the people), be a driving force, have support (from the government, the military, and the people), and be irreplaceable. For each capability, one needs to determine the existence of the essential conditions, resources and means that are required by that capability to be fully operative, and which of these, if any, represent critical vulnerabilities.

3. AGENT ARCHITECTURE

The architecture of Disciple-RKF includes the components from Figure 1, each implemented as a set of collaborative agents (Boicu et al., 2004). The core of the system is the learning agent shell, which has the following domain-independent components:

- A problem solving component based on the task reduction paradigm of problem solving.

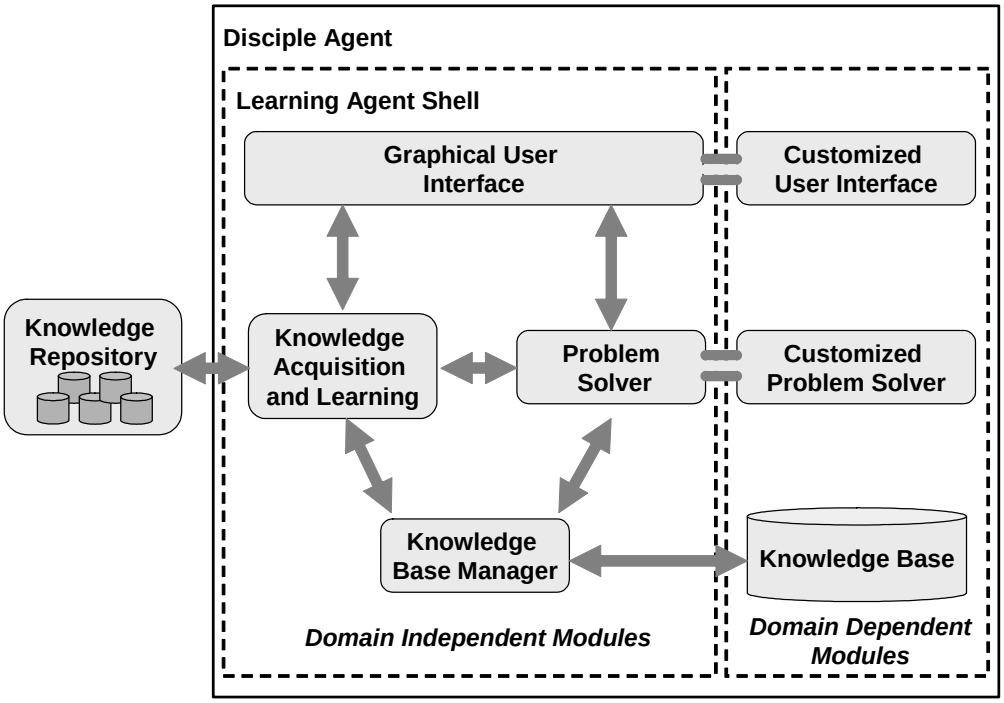

Figure 1: General architecture of a Disciple agent.

It includes a modeling agent that helps the user to express his/her contributions to the problem solving process, a mixed-initiative (step-by-step) problem solving agent, and an autonomous problem solving agent.

- A knowledge acquisition and learning component for acquiring and refining the knowledge of the agent, allowing a wide range of operations, including ontology import (from external knowledge repositories, such as CYC (Lenat, 1995)), user definition of knowledge base elements (through the use of editors and browsers), ontology learning, and rule learning.
- A knowledge base manager which controls the access to and the updates of the knowledge base. Each module of Disciple-RKF can access the knowledge base only through the functions of the knowledge base manager.
- A windows based, domain-independent, graphical user interface.

The three components in the right hand side of Figure 1 are the typical domain dependent components of a Disciple-RKF agent that was customized for a specific application, such as center of gravity analysis:

- A customized problem solving component that extends the basic task-reduction component in order to satisfy the specific problem solving requirements of the application domain.
- Customized graphical user interfaces which are built for the specific Disciple-RKF agent to allow the experts and the end users to communicate with the agent as close as possible to the way they communicate in their domains.
- The knowledge base of the Disciple-RKF agent that contains knowledge specific to the center of gravity analysis domain.

The next sections present the knowledge representation and the problem solving and learning methods implemented by these modules.

4. PROBLEM SOLVING

A Disciple-RKF agent performs problem solving tasks by using the task reduction paradigm (Nilsson 1971; Powell and Schmidt 1988). In this paradigm, a complex problem solving task is successively reduced to simpler tasks. Then the solutions of the simplest tasks are found and these solutions are successively combined into the solution of the initial task, as illustrated in the left hand side of Figure 2.

In the Disciple approach we have refined this general strategy so that it can be easily used both by the expert (when teaching the agent or when contributing to the joint problem solving process) and the agent (when solving a problem). We did this by introducing questions and

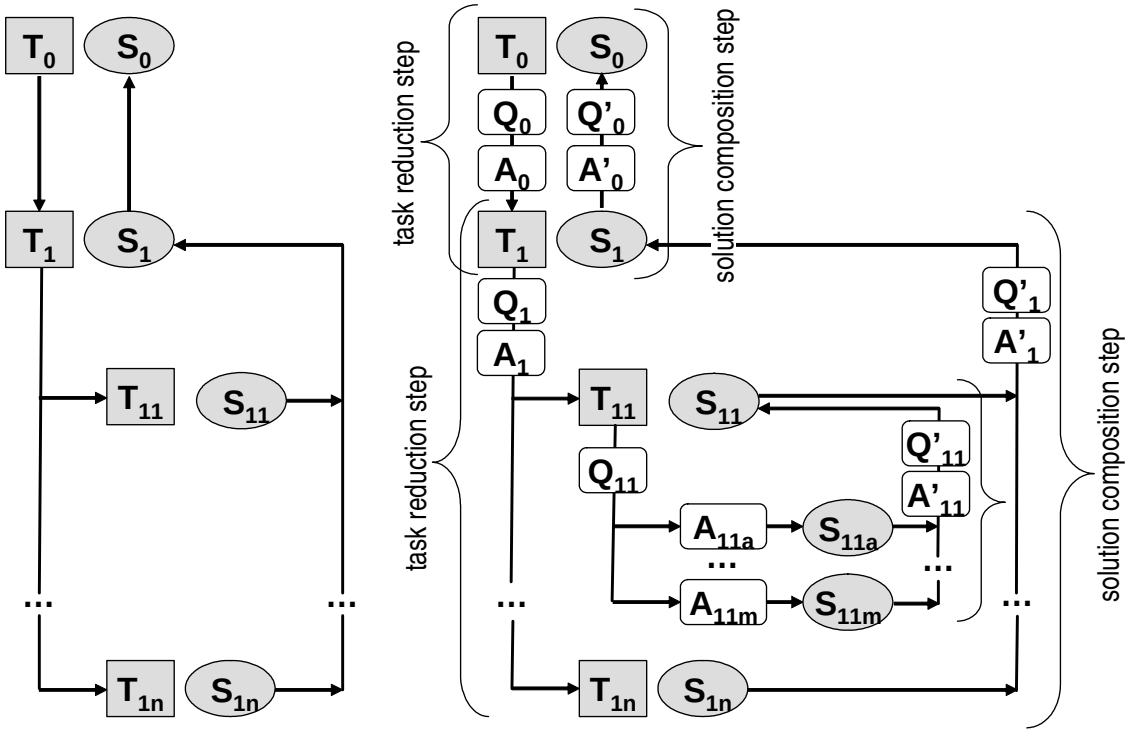

Figure 2: Problem-solving through task reduction and solution composition.

answers that guide the task reduction process, as illustrated on the right hand side of Figure 2 and discussed in more detail in (Bowman 2002). In this refined task reduction approach, finding a solution of a problem solving task (e.g. T_0 on the right hand side of Figure 2) becomes an iterative process where, at each step, the expert (or the agent, depending on who is doing the problem solving) looks for some relevant information for solving T_0 by asking a question Q_0. The answer A_0 identifies that piece of information and leads to the reduction of the current task to simpler tasks (e.g. T_1). Alternative questions correspond to alternative problem solving strategies. Multiple answers of a question (e.g. A_{11a} or A_{11m}) correspond to multiple solutions. Solution composition (e.g. the composition of S_{11a}, … , S_{11m} into S_{11}) is also guided by questions and answers.

Figure 8 illustrates the decomposition process with an example from the center of gravity analysis domain. In this example, the expert is showing the agent how to determine a center of gravity candidate for the Sicily 1943 scenario (World War II at the time of the invasion of the island of Sicily by the allied forces).

5. LEARNABLE KNOWLEDGE REPRESENTATION

The knowledge base of Disciple-RKF is structured into an object ontology and a set of task reduction rules and solution composition rules. The object ontology is a hierarchical representation of the objects from the application domain. It represents the different kinds of objects, the properties of each object, and the relationships existing between objects. The object ontology of Disciple-RKF for the center of gravity domain contains 355 object concepts. Moreover, a scenario such as the Sicily_1943 scenario used in this paper, is described with

around 900 facts (i.e. triplets of the form “object feature value”). A fragment of this object ontology for the center of gravity domain is shown in Figure 3.

The object ontology is incomplete. There are relevant concepts and instances from the application domain which are not represented. Moreover, the representation of a given concept, or instance, may be incomplete in the sense that it does not include all of its relevant properties and relationships. This object ontology will be extended by the agent during the problem solving and learning process (Boicu et al., 2003).

In addition to the hierarchy of instances and concepts illustrated in Figure 3, the object ontology also includes a hierarchy of features. In this hierarchy, for instance, the feature “has_as_head_of_government” is a subfeature of “has_as_political_leader,” which is a subfeature of “has_as_controlling_leader.” Each feature F is characterized by a domain and a range. The domain of F is a concept that represents all objects that may have the feature F. The range of F is a concept that represents all the possible values of F. The feature hierarchy for the

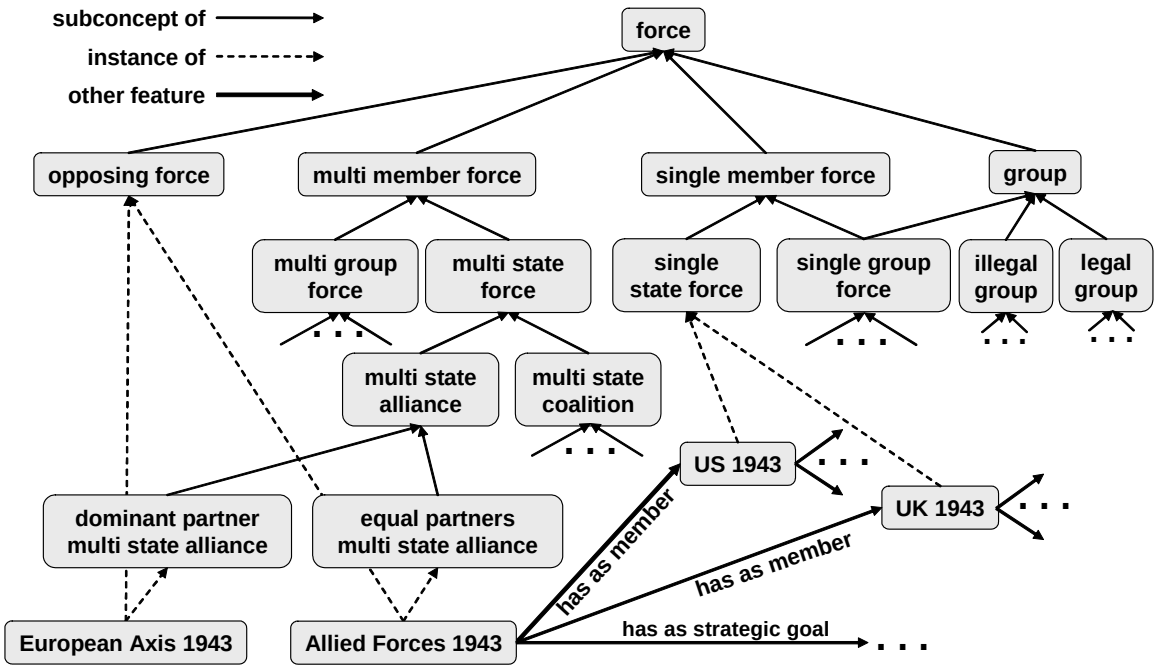

Figure 3. Fragment of the object ontology for the center of gravity domain.

center of gravity domain contains 193 feature definitions.

The object ontology plays a crucial role in Disciple, being at the basis of knowledge representation, user-agent communication, problem solving, knowledge acquisition and learning. Using the (object and feature) concepts from the object ontology, one can define more complex concepts as logical expressions involving these concepts. The basic representation unit (BRU) for such a concept has the form $\{?O_1, ?O_2, \ldots, ?O_n\}$, where each $?O_i$ has the structure indicated by [1].

$?O_i$	is	$concept_i$	[1]
	$feature_{i1}$	$?O_{i1}$	
	. . .		
	$feature_{in}$	$?O_{im}$	

$Concept_i$ is an object concept from the object ontology, a numeric interval, or a list of strings, and $?O_{i1} \ldots ?O_{im}$ are distinct variables from the set $\{?O_1, ?O_2, \ldots, ?O_n\}$. For example, the concept "the pair of entities $?O_1$ and $?O_2$, where $?O_1$ is an equal partner, multi-state alliance that has, as one of its members, $?O_2$, which is a single-state force" is represented by expression [2].

$?O_1$	is	equal_partners_multi_state_alliance	[2]
	has_as_member	$?O_2$	
$?O_2$	is	single_state_force	

In general, a concept may be a conjunctive expression of form [3], meaning that any instance of the concept satisfies BRU and does not satisfy BRU_1 and … and does not satisfy BRU_p.

$$BRU \ \& \ not \ BRU_1 \ \& \ \ldots \ \& \ not \ BRU_p \qquad [3]$$

However, instead of "not" we write "Except When," as illustrated in [4]. This concept represents "the pair of entities $?O_1$ and $?O_2$, where $?O_1$ is an equal partner multi-state alliance

that has, as one of its members, $?O_2$, which is single-state force, except when $?O_2$ is a single-state force with a minor military contribution."

$?O_1$	is	equal_partners_multi_state_alliance	[4]
	has_as_member	$?O_2$	
$?O_2$	is	single_state_force	

Except When

$?O_2$	is	single_state_force
	has_as_military_contribution	$?O_3$
$?O_3$	is	minor_military_contribution

The object ontology is at the basis of the generalization language for learning. For instance, a concept such as [2] may be generalized by replacing an object concept from its description (e.g. "equal_partners_multi_state_alliance") with a more general concept from the ontology (e.g. "multi_state _alliance"). Other generalization or specialization rules may be used to generalize or specialize such concepts as, for instance, dropping or adding an object feature or an Except When condition, generalizing a number to an interval, or an interval to a larger interval (Tecuci, 1998).

Partially learned concepts are represented as plausible version spaces (Tecuci, 1998), as illustrated in Figure 4. The plausible upper bound of this version space contains the two concepts shown in [5].

$?O_1$	is	multi_member_force	[5]
	has_as_member	$?O_2$	
$?O_2$	is	force	

and

$?O_1$	is	opposing_force
	has_as_member	$?O_2$
$?O_2$	is	force

Similarly, the plausible lower bound of this version space contains two concepts, one where $?O_1$ is a multi_state_alliance, and another one where $?O_1$ is an opposing_force. In the

current version of Disciple, the same features appear both in the upper bound and in the lower bound (such as "has_as_member" in the example from Figure 4).

The concept E_h to be learned (see Figure 4) is, ***as an approximation***, less general than ***one*** of the concepts from the plausible upper bound. E_h is also, again ***as an approximation***, more general than ***any*** of the concepts from the plausible lower bound. During learning, the two bounds converge toward one another through successive generalizations and specializations, approximating E_h better and better. This is different from the version spaces introduced by Mitchell (1978), where ***one*** of the concepts from the upper bound ***is always more general*** than the concept to be learned (and the upper bound is always specialized during learning), and ***any*** of the concepts from the lower bound ***is always less general*** than the concept to be learned (and the lower bound is always generalized during learning). The major difference is that the version spaces introduced by Mitchell (1978) are based on a complete representation space that includes the concept to be learned. On the contrary, the representation space for Disciple is based on an incomplete and evolving object ontology, as mentioned above. Therefore, Disciple addresses the more complex and more realistic problem of learning in the context of an evolving representation space.

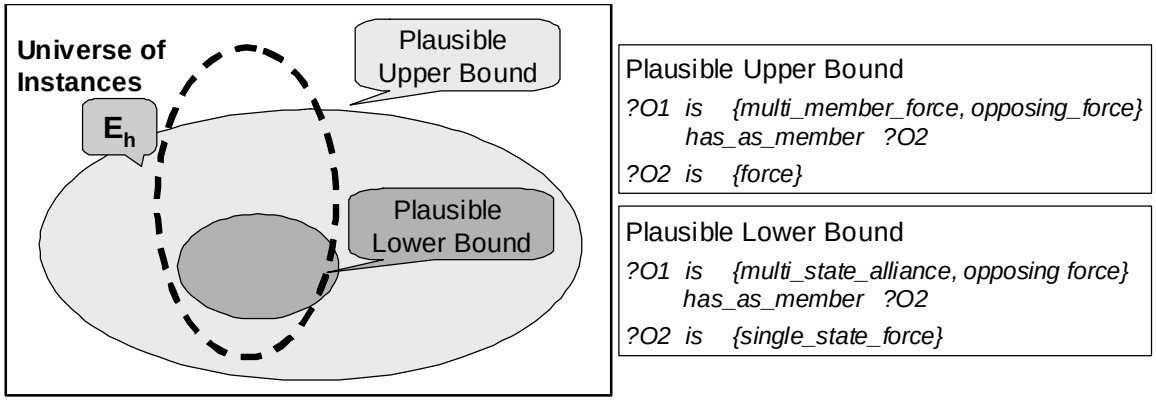

Figure 4: A plausible version space for a partially learned concept.

The notion of plausible version space is fundamental to the knowledge representation, problem solving, and learning methods of Disciple, as discussed below and in section 6. All the knowledge elements from the knowledge base are represented using this construct and are learned or refined by Disciple. For instance, Disciple-RKF learns general feature definitions from specific examples, the domains and the ranges of the partially learned features being represented as plausible version spaces.

The knowledge base of Disciple-RKF also contains tasks reduction rules and solution composition rules that are learned from specific examples of reductions or compositions. Figure 6 shows an example of a task reduction step and the task reduction rule learned from it. The rule is an IF-THEN structure that expresses under what condition a certain type of task may be reduced to a simpler subtask (or to several subtasks, in case of other rules). The rule is interpreted as follows: If the task to be solved is T_1, we are asking the question Q, and if the answer is A (or, equivalently, the applicability condition of the rule is satisfied), then we can reduce T_1 to T_{11}.

The rule in Figure 5 is a very simple one, with only a Main Condition. In general, however, in addition to a Main Condition, a learned rule may have several Except When Conditions (which should not be satisfied for the rule to be applicable), as well as positive and negative exceptions. Thus, in general, the condition of the rule is a concept of the form "BRU except-when BRU1 … except-when BRU_p" meaning that the rule may be applied for any instance of the condition concept. The rule shown in Figure 5 is only partially learned. Therefore, instead of a single applicability condition, it has a plausible version space for the exact condition to be learned. The knowledge base of Disciple-RKF contains 368 reduction rules, all learned by the agent. It also contains 269 composition rules.

6. MIXED-INITIATIVE MODELING, LEARNING AND PROBLEM SOLVING

The Disciple approach covers all the phases of agent development and use. First, a knowledge engineer works with a subject matter expert to develop an ontology for the application domain. They use the ontology import module (to extract relevant ontology elements from existing knowledge repositories) as well as the various ontology editors and browsers of Disciple-RKF. The result of this knowledge base development phase is an object ontology (see Figure 3) which is complete enough to be used as a generalization hierarchy for learning, allowing the expert to teach the Disciple agent how to solve problems, with limited assistance from a knowledge engineer. The teaching process is illustrated in Figure 6 and discussed in the following.

6.1. Rule Learning

The expert formulates an initial problem solving task, such as "Determine a center of gravity for the Sicily 1943 scenario," (see Figure 5) and shows the agent how to solve this task by using the task reduction paradigm described in section 4. The expert uses natural language, as if s/he would think aloud. S/he asks a question related to some piece of information which is relevant to solving the current task. The answer identifies that piece of information and leads the expert to reduce the current task to a simpler task (or, in other cases, to several simpler tasks). Figure 5 shows a sequence of task reduction steps. From each of these steps the agent learns a general task reduction rule. Table 1 and table 2 present the rule learning problem and method of Disciple-RKF. They will be illustrated in the following.

Let us consider the 4^{th} step from the task reduction tree in Figure 5, the same step also shown also on the left hand side of Figure 6. From this task reduction step, Disciple-RKF learned the task reduction rule shown in the right hand side of Figure 6.

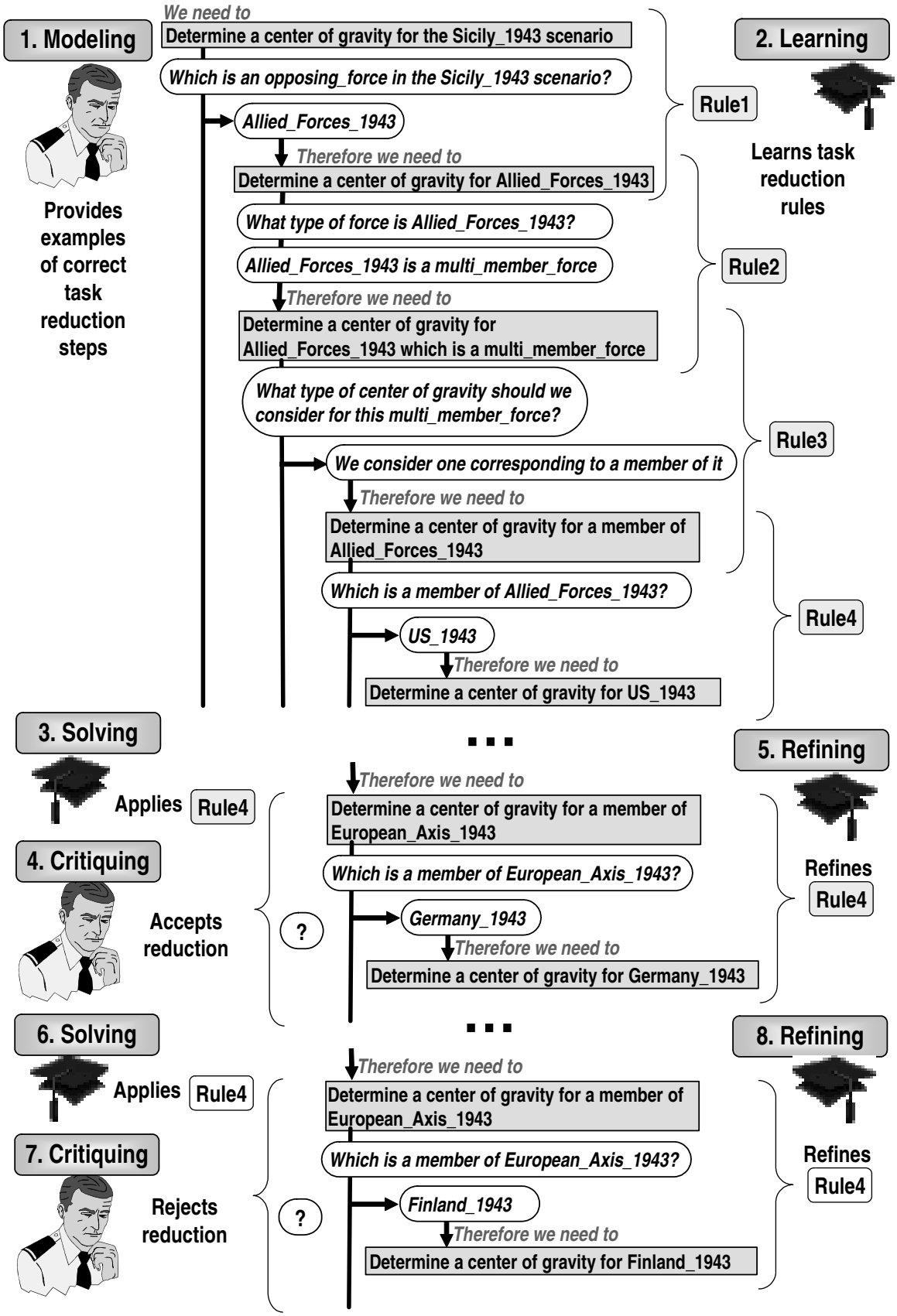

Figure 5: An illustration of mixed-initiative modeling, problem solving, and learning.

Table 1: The Rule Learning Problem.

GIVEN: • An example of a task reduction step. • A knowledge base that includes an object ontology and a set of task reduction rules. • A subject matter expert that understands why the given example is correct and may answer agent's questions. DETERMINE: • A plausible version space task reduction rule which is generalization of the specific task reduction step. • An extended object ontology (if needed for rule learning).

Rule learning is a mixed-initiative process between the expert (who knows why the reduction is correct and can help the agent to understand this) and the Disciple-RKF agent (that is able to generalize the task reduction example and its explanation into a general rule, by using the object ontology as a generalization language). This process is based on a communication protocol which takes into account that:

- It is easier for an expert to understand sentences in the formal language of the agent than it is to produce such formal sentences; and
- It is easier for the agent to generate formal sentences than it is to understand sentences in the natural language used by the expert.

The question and its answer from the task reduction step represent the expert's reason (or explanation) for performing that reduction. Because they are in natural language, the expert has to help Disciple-RKF "understand" them in terms of the concepts and features from the object ontology. Consider [6], the question and the answer from the example in Figure 6. Their meaning in the object ontology is expressed as in [7]. We call expression [7] the "explanation" of the example.

Table 2: The Rule Learning Method.

1. Explanation Generation Identify a formal explanation EX of why the example E is correct, through a mixed-initiative interaction with the subject matter expert. The explanation is an approximation of the meaning of the question and answer, expressed with the objects and the features from the object ontology. During the explanation generation process, new objects and features may be elicited from the expert and added to the object ontology. 2. Variable Generation Generate a variable for each instance, number and string that appears in the example and its explanation. Then use these variables, the example, and the explanation, to create an instance IC of the concept representing the applicability condition of the rule to be learned. This is the concept to be learned as part of rule learning. 3. Rule Generation Generate the tasks, question, and answer of the rule by replacing each instance or constant from the example E with the corresponding variable generated in step 2. Then generate the plausible version space of the applicability condition of the rule. The concept represented by this condition is the set of instances and constants that produce correct instantiations of the rule. The plausible lower bound of this version space is the minimally general generalization of IC determined in step 2, generalization which does not contain any instance. The plausible upper bound of this version space is the set of the maximally general generalizations of IC. 5. Rule Analysis If there is any variable from the THEN part of a rule which is not linked to some variable from the IF part of the rule, or if the rule has too many instances in the knowledge base, then interact with the expert to extend the explanation of the example and update the rule if new explanation pieces are found. Otherwise end the rule learning process.

"Which is a member of Allied_Forces_1943? US_1943" [6]

"Allied_Forces_1943 has_as_member US_1943" [7]

While a subject matter expert can understand the meaning of the above formal expression, s/he cannot easily define it for the agent because s/he is not a knowledge engineer. For instance, s/he would need to use the formal language of the agent. But this would not be enough, as the expert would also need to know the names of the potentially many thousands of concepts and features from the agent's ontology. Therefore, the agent will hypothesize plausible meanings of the question-answer pair by using simple natural language processing, analogical

reasoning with previously learned rules, and general heuristics, and will express them as explanation fragments. In general, an explanation fragment identified by the agent, such as [7], is a relationship (or a relationship chain) involving instances, concepts, and constants from the task reduction step and from the knowledge base. The agent will then propose these explanation pieces to the expert, ordered by their plausibility, so that the expert can select the ones the has the meaning of the question-answer pair. The expert may also help the agent to propose the right explanation pieces by proving hints, such as pointing to a relevant object that should be part of the explanation.

Using the example and its explanation, Disciple-RKF will generate the task reduction rule from the right hand side of Figure 6. First the agent will generate a variable for each instance, number, or string that appears in the example and its explanation. Then it will use these variables V to generalize the task reduction example E into an IF-THEN rule R, by replacing each instance or concept with the corresponding variable.

The next step in the rule learning process is to determine which are the instantiations of

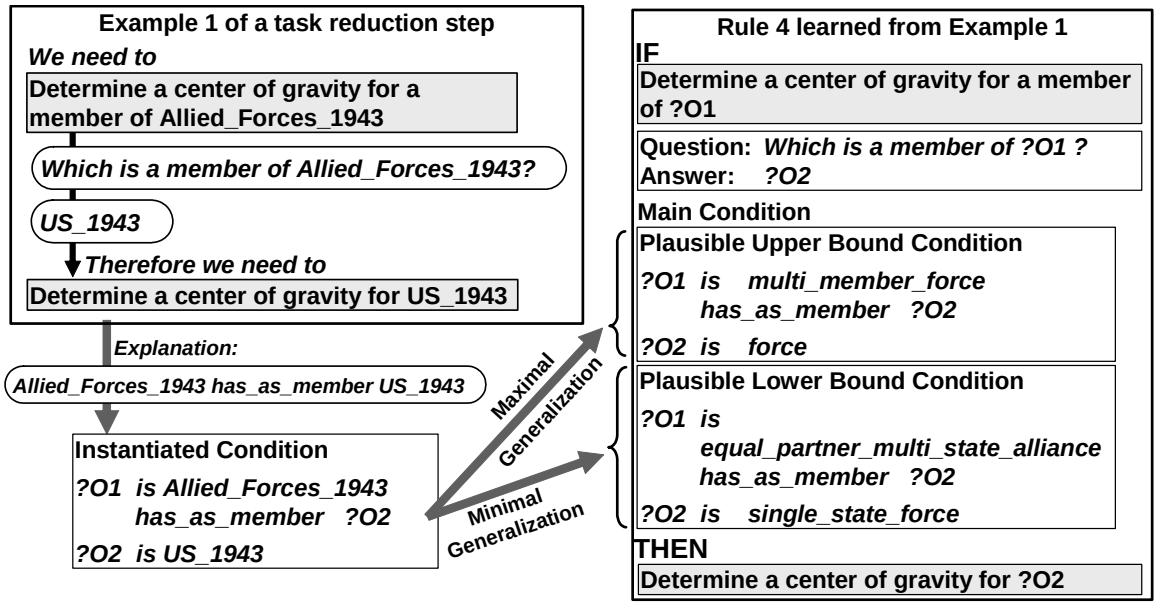

Figure 6: An example of a task reduction step and the rule learned from it.

the variables V that lead to correct task reduction steps. That is, we have to learn the concept that represents the set of instances of the rule's variables V for which the corresponding instantiation of the rule R is correct. We call this concept "the applicability condition of the rule R," and Disciple-RKF learns it by using a plausible version space approach. That is, it considers the set of all the applicability conditions that are consistent with the known examples and their explanations and it reduces this set as new examples and additional explanations are found. Moreover, as in the candidate elimination algorithm, this version space is represented by a plausible lower bound and by a plausible upper bound.

The initial plausible version space condition for the rule R is determined as follows. First one determines the instance of this condition, IC, corresponding to the initial example, as shown in the left hand side of Figure 6:

IC: ?O1 is Allied_Forces_1943 [8]
has_as_member ?O2
?O2 is US_1943

Notice that this condition includes the feature "has_as_member" from the explanation of the example. This is an essential feature of the objects from this example and, for the same reason as in the case of explanation-based learning (Mitchell et al., 1986, DeJong and Mooney, 1986), will significantly reduce the number of examples needed for learning.

Then one generalizes IC in two different ways to generate the two bounds of the version space. The plausible lower bound of the version space is the set of the least general generalizations of IC which includes no instance. The least general concepts from the object ontology that cover Allied_Forces_1943 are opposing_force and equal_partner_multi_state_alliance. However, Allied_Forces_1943 has the feature has_as_member, and therefore, any of its generalization should be in the domain of this feature,

which happens to be mulri_member_force. As a consequence, the set of the minimal generalizations of Allied_Forces_1943 is given by the following expression:

{opposing_force, equal_partner_multi_state_alliance} ∩ {multi_member_force} =

= {equal_partner_multi_state_alliance}

Similarly (but using the range of the has_as_member feature, which is force), one determines the set of the minimal generalizations of US_143 as {single_state_force}.

As a consequence, the plausible lower bound condition is:

PLB: ?O1 is equal_partner_multi_state_alliance [9]
has_as_member ?O2
?O2 is single_state_force

The reason the lower bound does not contain any instance is that the learned rule will be used by Disciple in other scenarios (such as Afghanistan_2001_2002), where the instances from Sicily_1943 do not exist, and Disciple-RKF would not know how to generalize them. On the other hand, we also do not claim that the concept to be learned is more general than the lower bound, as discussed in section x and illustrated in Figure 4.

Using a similar procedure (but considering the most general generalizations of the instances and constants from the example and its explanation), Disciple-RKF determines the plausible upper bound condition [10] and generates the rule from the right hand side of Figure 6.

PUB: ?O1 is multi_member_force [10]
has_as_member ?O2
?O2 is force

The last step of the rule learning process is to analyze the generated rule. For instance, the agent may determine that a variable from the THEN part of a rule is not linked to any variable from the IF part of the rule. This is indicative of a rule which was learned based on an incomplete explanation, causing the agent to reinitiate the explanation generation process.

Sometimes the missing explanation is so obvious to the expert that it is simply ignored, as in the following instance: “US_1943 has_as_government government_of_US_1943.” The agent will automatically select such an explanation if it provides a link to an unconstrained variable. Sometimes, even when all the rule’s variables are linked, the number of rule instances may still be very large. In such a case the agent will attempt to identify which variables are the least constrained and will attempt to further constrain them by proposing additional explanation pieces.

Notice that Disciple-RKF succeeded to learn a reasonable rule from only one example and its explanation, a rule that may be used by Disciple in the problem solving process. Indeed, it will apply the rule to reduce a current task if any of its plausible bound conditions is satisfied. Rule 4 has only one concept in its plausible upper bound, and also one concept in the plausible lower bound. In general, however, each bound may contain more than one concept, as discussed in section x and illustrated in Figure 4. We say that the plausible lower bound condition covers an example if **all** the alternative concepts from this bound covers the instances from the example. On the contrary, we say that the plausible upper bound condition covers an example if at least one of the alternative concepts from this bound covers the example.

6.2. Rule Refinement

As Disciple-RKF learns new rules from the expert, the interaction between the expert and Disciple evolves from a teacher-student interaction toward an interaction where both collaborate in solving a problem. During this mixed-initiative problem solving phase, Disciple learns not only from the contributions of the expert, but also from its own successful or unsuccessful problem solving attempts, which lead to the refinement of the learned rules. At the same time, Disciple may extend the object ontology with new objects and features.

Table 3: The Refinement Problem.

GIVEN: • A plausible version space task reduction rule R. • A positive or a negative example E of the rule (i.e. a correct or an incorrect task reduction step that has the same IF and THEN tasks as R). • A knowledge base that includes an object ontology and a set of task reduction rules. • A subject matter expert that understands why the task reduction step is correct or incorrect and can answer the agent's questions. DETERMINE: • A refined rule that covers the example if it is positive, or does not cover the example if it is negative. • An extended object ontology (if needed for rule refinement).

The rule refinement problem and methods are presented in Tables 3, 4 and 5. The result of the rule learning process described in Table 2 and illustrated above is a rule with a main plausible version space condition. During rule refinement, however, the rule may accumulate several "except when" plausible version space conditions. For that reason, Tables 3 and Table 4 assume that the rule R to be refined, based on the example E, has both a partially learned main condition and a partially learned "except when" condition. These methods are extended naturally when there are several such conditions. Figure 8 shows an abstract representation of the rule's conditions. The new (positive or negative) example of the rule (and of the rule's condition) may be situated in one of several relevant regions, as indicated in Figure 7. The way the rule is refined depends on the type of the example and the region in which it is situated, as described in Tables 3 and 4. In the following we will illustrate these methods.

Table 4: Rule refinement with a positive example.

1. If the positive example E is covered by ML and is not covered by XU (case 1 in Figure 7), then the rule does not need to be refined because the example is correctly classified as positive by the current rule. 2. If E is covered by MU, but it is not covered by ML and XU (case 2 in Figure 7), then minimally generalize ML to cover E and remain less general than MU. Remove also from MU the elements that do not cover E. 3. If E is not covered by MU (cases 3, 4, and 5 in Figure 7), or if E is covered by XL (cases 5, 6, and 7 in Figure 7), then keep E as a positive exception of the rule. 4. If E is covered by ML and XU, but it is not covered by XL (case 8 in Figure 7), then interact with the expert to find an explanation of the form: "The task reduction step is correct because I_i is C_i," where C_i is a concept from the ontology. If such an explanation is found, then XU is minimally specialized to no longer cover C_i. Otherwise, E is kept as positive exception. 5. If E is covered by MU and XU, but it is not covered by ML and XL (case 9 in Figure 7), then minimally generalize ML to cover E and remain less general than MU. Also remove from MU the elements that do not cover E. Then continue as in step 4.

As indicated in Figure 5, Disciple-RKF applied Rule 4 to reduce the task "Determine a center of gravity for a member of European_Axis_1943," generating an example that is covered by the plausible upper bound condition of the rule. This reduction was accepted by the expert as correct. Therefore, Disciple generalized the plausible lower bound condition to cover it. For instance, European_Axis_1943 is a multi_member_force, but it is not an equal_partner_multi_state_alliance. It is a dominant_partner_multi_state_alliance dominated by Germany_1943. As a consequence, Disciple-RKF automatically generalizes the plausible lower bound condition of the rule to cover this example. The refined rule is shown in the left-hand side of Figure 8. This refined rule is then generating the task reduction from the bottom part of Figure 5. Although this example is covered by the plausible lower bound condition of the rule, the expert rejects the reduction as incorrect. This shows that the plausible lower bound condition is

Table 5: Rule refinement with a negative example.

1. If the negative example E is covered by ML and it is not covered by XU (case 1 in Figure 7), then interact with the subject matter expert to find an explanation of why E is a wrong task reduction step. If an explanation EX is found, then generate a new Except When plausible version space condition and add it to the rule. Otherwise, keep E as a negative exception. 2. If E is covered by MU but it is not covered by ML and by XU (case 2 in Figure 7) then interact with the expert to find an explanation of why E is a wrong task reduction step. If an explanation EX is found and it has the form "I_i is not a C_i," where C_i is a concept covered by MU, then specialize MU to be covered by C_i. Otherwise, if another type of explanation EX is found then learn a new Except When condition based on it, and add this condition to the rule. 3. If E is not covered by MU (cases 3, 4, 5 in Figure 7), or it is covered by XL (cases 5, 6, 7 in Figure 7), then the rule does not need to be refined because the example is correctly classified as negative by the current rule. 4. If E is covered by ML and XU but it is not covered by XL (case 8 in Figure 7), or E is covered by MU and XU but it is not covered by ML and XL (case 9 in Figure 7), then minimally generalize XL to cover E and specialize XU to no longer include the concepts that do not cover E.

not more general than the concept to be learned (as it would have been the case in the classical candidate elimination algorithm) and it would need to be specialized.

This rejection of the reduction proposed by Disciple-RKF initiates an explanation generation interaction during which the expert will have to help the agent understand why the reduction step is incorrect. The explanation of this failure is that Finland_1943 has only a minor military contribution to European_Axis_1943 and cannot, therefore, provide the center of gravity of this alliance. The actual failure explanation (expressed with the terms from the object ontology) has the form:

"Finaland_1943 has_as_military_contribution military_contribution_of_Finaland_1943 is minor_military_contribution"

Based on this failure explanation, Disciple_RKF generates a plausible version space except when condition and adds it to the rule, as indicated in the right hand side of Figure 8. In

the future, this rule will only apply to situations where the main condition is satisfied and the except when condition is not satisfied.

Notice that the addition of the except when condition specializes both bounds of the applicability condition of the rule. Other types of failure explanations may lead to different modifications of Rule 4. For instance, the failure explanation may have had the form "Finland_1943 is not a major_ally," if "major_ally" would have been part of the object ontology (which it is not). In such a case, Disciple-RKF would not add an except when condition but it would specialize both bounds of the main condition to cover only "major_ally." Yet another possibility is to find an additional feature of the positive examples of the rule which is not a feature of the current negative example. This feature would then be added to the corresponding object from the main condition (both in the upper bound and in the lower bound). Additional negative examples may lead to additional except when conditions and specializations of the main condition.

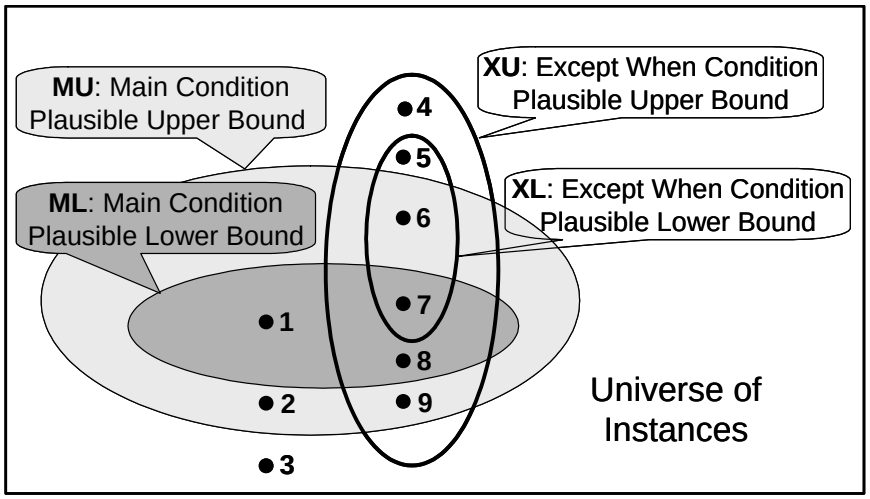

Figure 7: Possible regions for a new (positive or negative) example of a rule.

It may be the case that the actual explanation of an example contains elements that are not part of the object ontology. In such situations, Disciple-RKF elicits these elements from the expert and adds them to the ontology as new concepts or new features. Thus the learning process is performed in an evolving representation space in which the object ontology (which is used as the generalization hierarchy for learning) may be modified at any time. If the ontology is modified, the learned rules may no longer be correct. Therefore, the rules need to be relearned. This can be automatically done if the system keeps the examples and the explanations from which the rules were learned. However, different examples and explanations of a rule contain instances that existed in different scenarios (e.g. World War II or Afghanistan 2001-2002).

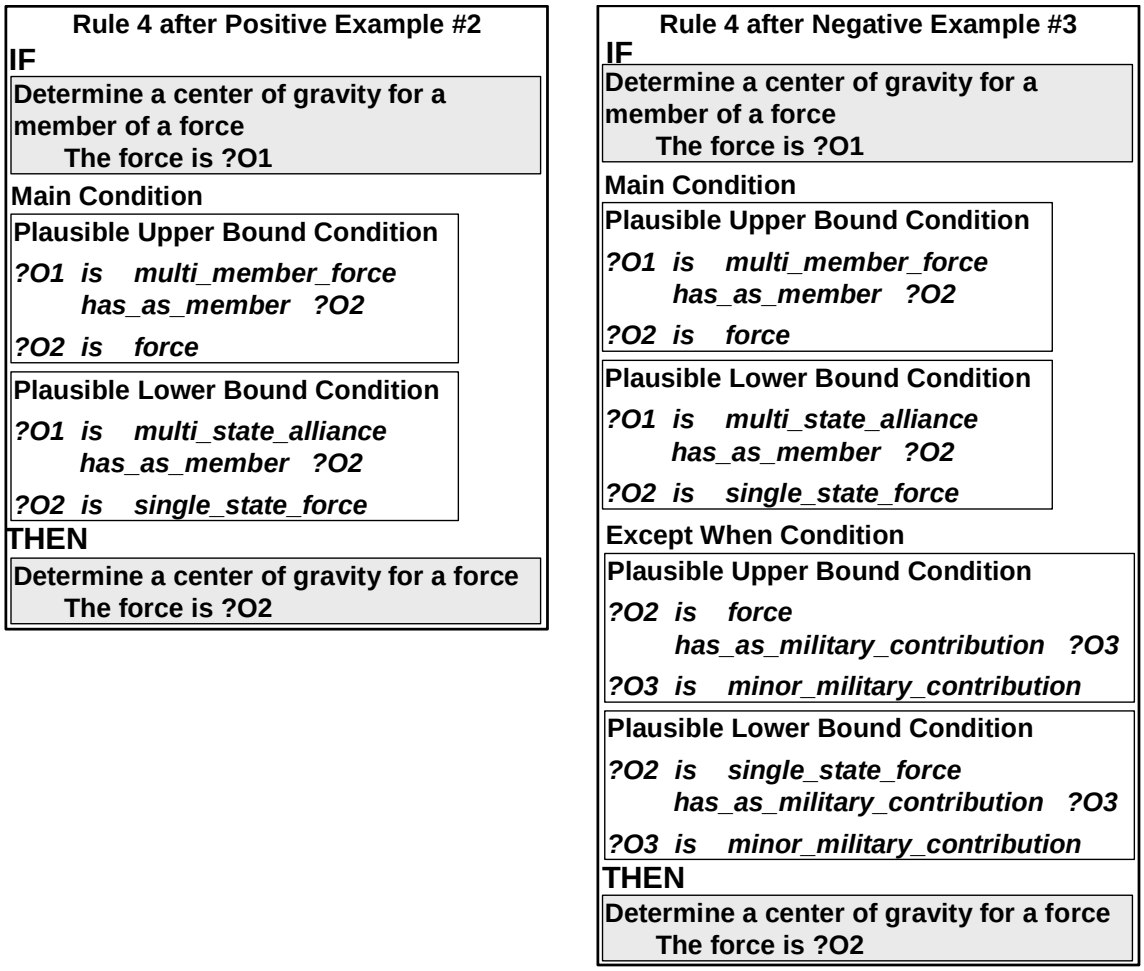

Figure 8: Refinements of Rule 4

Therefore, Disciple keeps minimal generalizations of the examples and explanations that do not contain any instance and use these minimally generalized examples and explanations to regenerate the rules when the object ontology changes

7. DEPLOYMENT AND AGENT TRAINING EXPERIMENTS

Successive versions of Disciple-RKF were used in two courses at the US Army War College since 2001, while still under research and development, becoming part of their regular syllabus (Tecuci et al., 2002a; 2002b; 2004a). For the "Case Studies in Center of Gravity Analysis" course, which is offered twice a year, Disciple-RKF was taught based on the expertise of Prof. Jerome Comello, the course's instructor. Then the students used Disciple-RKF as an intelligent assistant that helped them to develop a center of gravity analysis of a war scenario. *This demonstrates that the Disciple approach can be used to develop agents that have been found to be useful for a complex military domain.*

In the second course, "Military Applications of Artificial Intelligence," which was offered once a year between 2001 and 2003, the students (who were subject matter experts at the rank of lieutenant colonel or colonel) taught personal Disciple-RKF agents their own expertise in center of gravity determination and then evaluated both the developed agents and the development process. Figure 9 presents the most complex of these experiments, performed in Spring 2003 (Tecuci et al., 2004b).

Before starting the experiment, Disciple-RKF was trained to identify leaders as center of gravity candidates. The knowledge base of this agent contained the definitions of 432 concepts and features and 18 task reduction rules. However, the agent had no knowledge of how to test the identified candidates. We then performed a joint domain analysis and ontology development

with all the experts by considering the example of testing whether Saddam Hussein, in the Iraq 2003 scenario, would have all the required critical capabilities to be the center of gravity for Iraq. Based on this domain analysis, we have extended the ontology of Disciple-RKF with the definition of 37 new concepts and features identified with the help of the experts.

The 13 subject matter experts from the class were then grouped into five teams (of 2 or 3 experts each), and each team was given a copy of the extended Disciple-RKF agent. Next, each team trained its agent to test whether a leader has one or two critical capabilities, as indicated in Figure 9. For instance, Team 1 trained its agent how to test whether a leader had the critical capabilities of staying informed and being irreplaceable. The training was done based on three scenarios (Iraq 2003, Arab-Israeli 1973, and War on Terror 2003) and the experts teaching Disciple-RKF how to test each strategic leader from these scenarios. As a result of the training

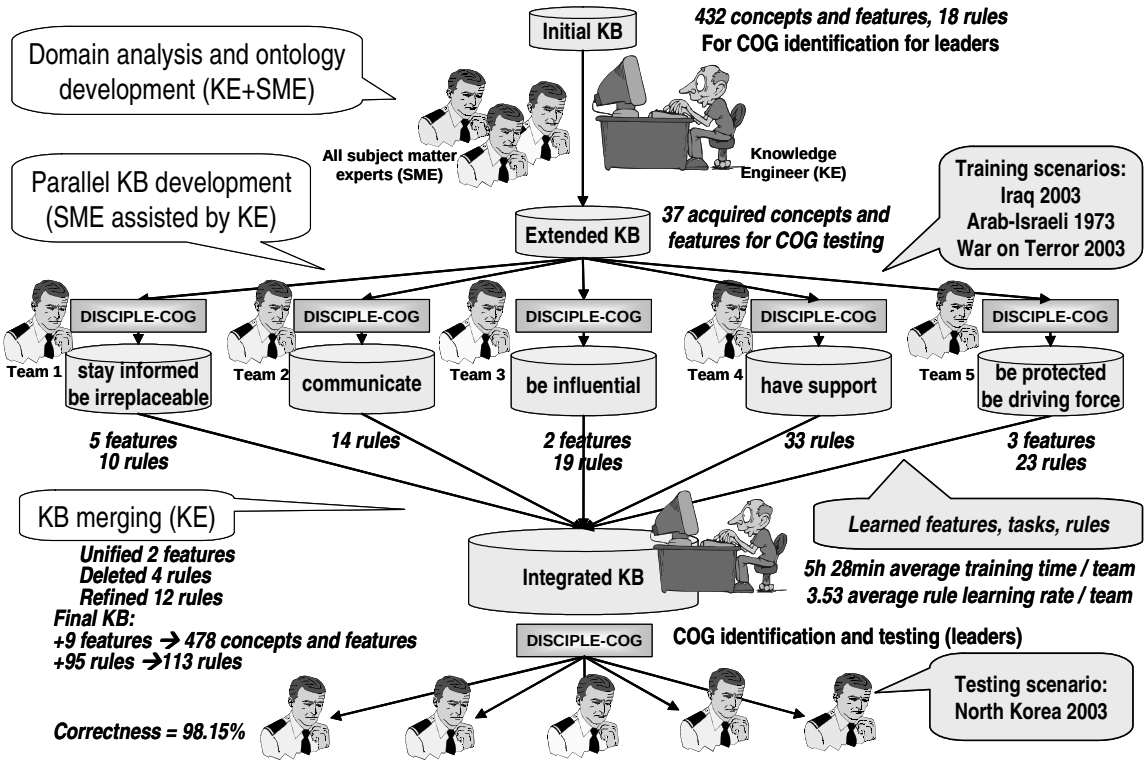

Figure 9. Experiment of rapid knowledge base development by subject matter experts.

performed by the experts, the knowledge base of each Disciple-RKF agent was extended with new object features and rules, as indicated in Figure 9. For instance, the knowledge base of the agent trained by Team 1 was extended with 5 features and 10 task reduction rules. The average training time per team was 5 hours and 28 minutes and the average rule learning rate per team was 3.53 rules/hour. This included time spent in all the agent training activities (i.e., specifying the three training scenarios, modeling experts' reasoning, rule learning, mixed-initiative problem solving, and rule refinement).

After the training of the 5 Disciple-RKF agents, their knowledge bases were merged by a knowledge engineer, who used the knowledge base merging tool of Disciple-RKF. The knowledge engineer also performed a general testing of the integrated knowledge base, in which he included the 10 features and 99 rules learned. During this process, two semantically equivalent features were unified, 4 rules were deleted, and 12 other rules were refined by the knowledge engineer. The other 8 features and 83 rules learned were not changed. Most of the modifications were done to remove rule redundancies or to specialize overly general rules.

Next, each team tested the integrated agent on a new scenario (North Korea 2003) and was asked to judge the correctness of each reasoning step performed by the agent, but only for the capabilities for which that team performed the training of the agent. The result was 98.15% correctness. Moreover, at the end of the experiment, 7 experts strongly agreed, 4 agreed, 1 was neutral, and 1 disagreed with the statement "*I think that a subject matter expert can use Disciple to build an agent, with limited assistance from a knowledge engineer*".

This is the first time that such an agent training and knowledge bases merging experiment has been performed, and the obtained results support the claim that the Disciple learning-based approach can be used to develop complex knowledge-based agents.

8. RELATED RESEARCH AND DISCUSSION

To our knowledge, Disciple-RKF, as a whole, is quite unique, both in the field of Machine Learning and in the field of Knowledge Acquisition, and we are not aware of any other system that is similar in terms of capabilities and methods to achieve them. However, some of its aspects can be compared with other systems. For instance, from the point of view of its general approach to learning, Disciple-RKF is mostly related to the older work on apprenticeship learning (DeRaedt 1991; Mahadevan et al. 1993; Mitchell et al., 1985; Tecuci 1988; Wilkins 1990). However, none of these older systems have the level of maturity and scalability required for practical applications. Nor are they integrated tools for building end-to-end knowledge-based agents, going through all the phases of agent development, including modeling expert's reasoning, ontology development and learning, and mixed-initiative problem solving and learning. The type of user was not a concern of these older apprentice systems, while Disciple-RKF was specially developed to allow a subject matter expert who does not have prior computer science or knowledge engineering experience, to train it. This was achieved by using multiple interactions with the expert and using multistrategy learning methods that incorporate learning from examples, learning from explanations, and learning by analogy and experimentation.

Because the idea of version spaces plays such a crucial role in Disciple-RKF, we can also compare it with other systems based on this general approach, such as (Mitchell et al., 1983; Hirsh, 1989; Sebag, 1996). We are again not aware of any system based on the version space representation that can deal with complex applications. One cause of this is the combinatorial explosion of the bounds of the version space. Disciple is able to avoid this combinatorial explosion because of the use of the explanations that identify the essential features of the

concepts to be learned. Another difference is that Disciple-RKF can learn an approximation of a concept even if the actual concept is not representable in its language, and can learn concepts with exceptions. One of our future research direction with respect to the plausible version space representation is the use of disjunctions and negations in the plausible version spaces of individual variables, such as ?O1 in Figure x, along the lines already investigated by Boicu (2002). This will allow Disciple to learn the best approximations of the concepts, when they are not representable.

Finally, Disciple-RKF can be compared with the tools for building knowledge-based systems, such as CYC (Lenat, 1995), EXPECT (Kim and Gil, 1999), or Loom (MacGregor, 1991). These tools make very little use of machine learning, while learning plays a crucial role in Disciple-RKF. Although Disciple-RKF has not been directly compared with these tools on the same problems, two previous versions of Disciple, Disciple-WA (Tecuci et al., 1999) and Disciple-COA (Boicu et al., 2000; Tecuci et al., 2001), have been compared with these tools, as part of the DARPA's High Performance Knowledge Bases program (Cohen et al., 1998), and demonstrated both higher knowledge acquisition rates, and better performance of the resulting systems. Rule 4 in Figure 8 helps explain why Disciple performed so well. In the case of the other systems, a knowledge engineer needed to manually define and debug the problem solving rules. With Disciple, the domain expert (possibly assisted by a knowledge engineer) needs only to define specific reductions like those in Figure 5, because Disciple will learn and refine the corresponding rules.

There are, however, many ways in which Disciple-RKF can still be considerably improved. For instance, one needs to develop more powerful methods for helping the expert to express his or her reasoning process using the task reduction paradigm, along the path opened by

the Modeling Advisor described in (Boicu, 2002). Our agent training experiments have also revealed that the mixed-initiative learning methods of Disciple-RKF could be significantly empowered by developing the natural language processing capabilities of the system. More powerful ontology learning methods are also needed (Stanescu et al., 2003). Finally, because the expert who teaches Disciple-RKF has no formal training in knowledge engineering, the knowledge pieces learned by the agent and the knowledge base itself will not be optimally represented, and will require periodic revisions by the knowledge engineer. Examples of encountered problems with the knowledge base are semantic inconsistencies within a rule, and the violation of certain knowledge engineering principles. It is therefore necessary to develop mixed-initiative knowledge base reformulation and optimization methods to identify and correct such problems in the knowledge base.

ACKNOWLEDGEMENTS

This research was done in the Learning Agents Center of George Mason University and was sponsored by several US Government agencies, including the Defense Advanced Research Projects Agency, the Air Force Research Laboratory, the Air Force Office of Scientific Research, and the US Army War College.

REFERENCES

Boicu C., Tecuci G., Boicu M., Marcu D. 2003. "Improving the Representation Space through Exception-Based Learning," in *Proceedings of the 16th International FLAIRS Conference* (FLAIRS-2003), *Special Track on Machine Learning*, May 2003, Key West, Florida. AAAI Press, Menlo Park, CA, pp. 336-340.

Boicu, M. 2002. Modeling and Learning with Incomplete Knowledge, PhD dissertation. George Mason University, Fairfax, Virginia, USA.

Boicu M., Tecuci G., Marcu D., Bowman M., Shyr P., Ciucu F., and Levcovici C. 2000. Disciple-COA: From Agent Programming to Agent Teaching, *Proceedings of the Seventeenth International Conference on Machine Learning*, Stanford, CA, Morgan Kaufmann.

Boicu M., Tecuci G., Stanescu B., Marcu D., Barbulescu M., Boicu C. 2004. "Design Principles for Learning Agents," in *Proceedings of AAAI-2004 Workshop on Intelligent Agent Architectures: Combining the Strengths of Software Engineering and Cognitive Systems*, July 26, San Jose, AAAI Press, Menlo Park, CA.

Bowman, M. 2002. A Methodology for Modeling Expert Knowledge that Supports Teaching Based Development of Agents, PhD dissertation. George Mason University, Fairfax, Virginia, USA.

Clausewitz, C.V. 1832. *On War*, translated and edited by M. Howard and P. Paret. Princeton, NJ: Princeton University Press, 1976.

Cohen P., Schrag R., Jones E., Pease A., Lin A., Starr B., Gunning D., & Burke M. 1998. The DARPA High-Performance Knowledge Bases project, *AI Magazine, 19,* 25-49.

DeJong, G. and Mooney, R.J. 1986. "Explanation-Based Learning: An Alternative View," *Machine Learning*, Vol. 1, pp. 145-176.

De Raedt L. 1991. Interactive Concept Learning. *Ph.D. Thesis*, Catholic University of Leuven.

Giles, P.K., and Galvin, T.P. 1996. *Center of Gravity: Determination, Analysis and Application*. CSL, U.S. Army War College, PA: Carlisle Barracks.

Hirsh, H. 1989. "Incremental Version-space Merging: A General Framework for Concept Learning," *Doctoral dissertation,* Stanford University.

Kim, J. & Gil, Y. 1999. Deriving expectations to guide knowledge base creation. *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Conference on Innovative Application of Artificial Intelligence* (pp. 235-241). Menlo Park, CA: AAAI Press.

Lenat, D.B. 1995. CYC: a large-scale investment in knowledge infrastructure. *Communications of Association for Computing Machinery, 38,* 33-38.

Mahadevan, S., Mitchell, T., Mostow, J., Steinberg, L. & Tadepalli, P. 1993. An apprentice based approach to knowledge acquisition, *Artificial Intelligence*, 64, 1-52.

Michalski R.S. and Tecuci G. (eds). 1994. "Machine Learning: A Multistrategy Approach," vol. IV, 782 pages, Morgan Kaufmann, San Mateo.

Mitchell, T.M. 1978. "Version Spaces: an Approach to Concept Learning," *Doctoral Dissertation,* Stanford University.

Mitchell, T.M., Keller, T. and Kedar-Cabelli, S. 1986. "Explanation-Based Generalization: A Unifying View," *Machine Learning,* Vol. 1, pp. 47-80.

Mitchell T., Mahadevan S. & Steinberg L. 1985. LEAP: a Learning Apprentice System for VLSI Design, Proc. IJCAI-85, Los Angeles, 573-580.

Mitchell, T.M., Utgoff P.E., Banerji R. 1983. Learning by Experimentation: Acquiring and Refining Problem-Solving Heuristics, in Michalski, R., Carbonell, J., and Mitchell, T. (eds.), Machine Learning, Vol. I, Morgan Kaufmann, San Mateo, CA.

MacGregor, R. 1991. The evolving technology of classification-based knowledge representation systems. In Sowa, J. ed. *Principles of Semantic Networks: Explorations in the Representations of Knowledge,* (pp. 385-400). San Francisco, CA: Morgan Kaufmann.

Nilsson, N.J. 1971. *Problem Solving Methods in Artificial Intelligence,* NY: McGraw-Hill.

Powell G.M. and Schmidt C.F. 1988. A First-order Computational Model of Human Operational Planning, CECOM-TR-01-8, US Army CECOM, Fort Monmouth, New Jersey, August.

Sebag M. 1996. Delaying the Choice of Bias: A Disjunctive Version Space Approach. In Saitta, L. (editor) Machine Learning – Proceedings of the Thirteenth International Conference (ICML' 96), pp. 444-452. San Francisco, California: Morgan Kaufmann Publishers, Inc.

Stanescu B., Boicu C., Balan G., Barbulescu M., Boicu M., Tecuci G. 2003. Ontologies for Learning Agents: Problems, Solutions and Directions. In *Proceedings of the IJCAI-03 Workshop on Workshop on Ontologies and Distributed Systems*, 75-82. Acapulco, Mexico, AAAI Press, Menlo Park, CA.

Strange, J. 1996. Centers of Gravity & Critical Vulnerabilities: Building on the Clausewitzian Foundation So That We Can All Speak the Same Language. Quantico, Virginia, USA, Marine Corps University.

Tecuci, G. 1988. Disciple: A Theory, Methodology and System for Learning Expert Knowledge, *Thèse de Docteur en Science,* University of Paris-South.

Tecuci, G. 1998. *Building intelligent agents: an apprenticeship multistrategy learning theory, methodology, tool and case studies*. London: Academic Press.

Tecuci, G., Boicu, M., Wright, K., Lee, S.W., Marcu, D. & Bowman, M. 1999. An integrated shell and methodology for rapid development of knowledge-based agents, *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Conference*

on Innovative Application of Artificial Intelligence (pp. 250-257). Menlo Park, CA: AAAI Press.

Tecuci G., Boicu M., Bowman M., and Marcu D., with a commentary by Burke M. 2001. An Innovative Application from the DARPA Knowledge Bases Programs: Rapid Development of a High Performance Knowledge Base for Course of Action Critiquing. *AI Magazine,* 22, 2. AAAI Press, Menlo Park, California, 43-61.

Tecuci G., Boicu M., Marcu D., Stanescu B., Boicu C., Comello J., Lopez A., Donlon J., Cleckner W. 2002a. "Development and Deployment of a Disciple Agent for Center of Gravity Analysis," in *Proceedings of the Eighteenth National Conference of Artificial Intelligence and the Fourteenth Conference on Innovative Applications of Artificial Intelligence,* AAAI-02/IAAI-02, pp. 853 - 860, Edmonton, Alberta, Canada, AAAI Press/The MIT Press. *Deployed Application Award.*

Tecuci G., Boicu, M., Marcu, D., Stanescu, B., Boicu, C., and Comello, J. 2002b. Training and Using Disciple Agents: A Case Study in the Military Center of Gravity Analysis Domain. AI Magazine 23(4) 51–68.

Tecuci G., Aha D., Boicu M., Cox M., Ferguson G., and Tate A. (eds). 2003. *Proceedings of the IJCAI-03 Workshop on Mixed-Initiative Intelligent Systems,* Acapulco, Mexico, August, 143 pg.

Tecuci G., Boicu M., Marcu D., Stanescu B., Boicu C., Barbulescu M., 2004a. "A University Research Group Experience with Deploying an Artificial Intelligence Application in the Center of Gravity Analysis Domain," in *Proceedings of AAAI-2004 Workshop on Fielding Applications of Artificial Intelligence,* July 25, San Jose, AAAI Press, Menlo Park, CA.

Tecuci G., Boicu M., Marcu D., Stanescu B., Boicu C., Barbulescu M. 2004b. "Parallel Knowledge Base Development by Subject Matter Experts" in *Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management*, EKAW 2004, 5-8th October 2004 - Whittlebury Hall, Northamptonshire, UK, Springer-Verlag.

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