In AI Magazine, 22, 2, 2001, pp. 43-61. AAAI Press, Menlo Park, California, 2001.

An Innovative Application from the DARPA Knowledge Bases Programs: Rapid Development of a Course of Action Critiquer

Gheorghe Tecuci, Mihai Boicu, Mike Bowman and Dorin Marcu
with a preface by Murray Burke

DARPA's HPKB and RKF Programs in Knowledge Base Development

In his invited talk at the 1993 National Conference on Artificial Intelligence, Edward Feigenbaum compared the technology of a knowledge based computer system with a tiger in a cage. Rarely does a technology arise that offers such a wide range of important benefits. Yet, this technology is still very far from achieving its potential. This tiger is in a cage and to free it the Artificial Intelligence research community must understand and remove the bars of the cage.

We now know that a knowledge based-system needs a great deal of knowledge in order to be truly useful. However, building a large and high performance knowledge base is still a very long, inefficient and error-prone process. Responding to this problem, DARPA has sponsored a sequence of two programs for the development of the second generation of knowledge-based systems science and technology: the High Performance Knowledge Bases program (HPKB FY97-99) and the Rapid Knowledge Formation program (RKF FY00-03). The goal of the HPKB Program was to produce the technology needed to enable system developers to rapidly construct large knowledge bases that provide comprehensive coverage of topics of interest, are reusable by multiple applications with diverse problem-solving strategies, and are maintainable in rapidly changing environments. The tasks of knowledge representation and acquisition are too difficult to start from scratch each time a new knowledge base needs to be built. Therefore, this program supported the development of methods for creating knowledge bases by selecting, composing, extending, specializing, and modifying components from a library of reusable ontologies, common domain theories, and generic problem-solving strategies. In addition, it supported the development of methods for rapidly extracting knowledge from natural language texts and the World Wide Web, and for knowledge acquisition from subject matter experts (SMEs). An important emphasis of the HPKB program was the use of challenge problems. These are complex, innovative military applications of Artificial Intelligence, intended to focus the research and development efforts and measure the effectiveness of alternative technical approaches. The participants collaborated in the development of knowledge-based systems for solving these challenge problems. These systems were subject of intensive annual evaluations during which the completeness and correctness of the developed knowledge bases were measured, as well as the time required to build the knowledge bases and the ease of modifying them to assimilate new or changed knowledge. The challenge problems for the first part of the HPKB program and the corresponding evaluation results were presented in the Winter 1998 issue of the AI magazine. The second part of the HPKB program was based on even more complex challenge problems. One challenge problem was an extension of the year one Crisis Management problem, requiring rapid development of a large knowledge base containing broad but relatively shallow knowledge (such as general knowledge of countries, politics, geopolitical events, economics) necessary to discover and understand information about nascent and emerging crises, from a wide range of potential information sources. Two teams, one composed of Teknowledge, Cycorp, Textwise, and Northwestern University, and the other composed of SAIC, SRI, MIT, Stanford University and Northwestern University, developed two end-to-end integrated systems that were evaluated by IET, the challenge problem developer, in the summer of 1999. They both demonstrated high performance through knowledge reuse and semantic integration, and created a significant amount of reusable knowledge. The other challenge problem required the rapid development of knowledge bases containing comprehensive battlefield knowledge (e.g., terrain characteristics, force structures, military organizations, troop movements, military strategy). The problem was to assess various aspects of military courses of action (COA), such as their viability, their correctness, and their strengths and weaknesses with respect to the principles of war and the tenets of army operations, to justify the assessments made and to propose improvements. This challenge problem was solved by developing a complex end-to-end system integrating complementary technologies developed by different research groups. Teknowledge, Northwestern University, and the University of Edinburgh's AIAI developed the translation and fusion module that interpreted and combined the information included in the COA sketch and the COA statement, and generated an internal representation of the input COA, based on a CYC ontology. Four critiquers, each developed by a different team (a joint team from Teknowledge and Cycorp, the Disciple team from GMU, the Expect team from ISI/USC, and the Loom team from ISI/USC) shared the generated representation of the COA and critiqued different aspects of it. The answers from each critiquer were displayed by a solution viewer developed by SAIC. The integrated system and its individual components were evaluated by Alphatech as exhibiting performance at the level of a subject matter expert.

The HPKB Program emphasized the development of very large knowledge bases by knowledge engineers, demonstrating the utility of large knowledge bases and the feasibility of large-scale reuse. It has produced reusable knowledge libraries including an upper ontology and middle theories for crisis and battlefield reasoning. The follow-on RKF Program emphasizes the development of knowledge bases directly by the domain experts. Its central objective is to enable distributed teams of SMEs to enter and modify knowledge directly and easily, without the need for prior knowledge engineering experience. The emphasis is on content and the means of rapidly acquiring this content from individuals who possess it with the goal of gaining a scientific understanding of how ordinary people can work with formal representations of knowledge. Therefore, the Program's primary requirement is the development of functionality enabling SMEs to understand the contents of a knowledge base, enter new theories, augment and edit existing knowledge, test the adequacy of the knowledge base under development, receive explanations of theories contained in the knowledge base, and detect and repair errors in content. Because of the complexity of these tasks, the approaches developed in RKF exploit the synergies among complementary AI technologies, such as natural language discourse processing, problem solving and learning by analogy, and common sense reasoning.

RKF is organized in a manner similar to HPKB, with challenge problems administered by an evaluation contractor (IET assisted by Veridian/PSR and George Mason University) as a basis for formal evaluation of the technology provided by the developers. There are two integrated teams and several component technology developers that contribute to one or to both of them. One team is coordinated by Cycorp and includes research groups from ISI/USC, University of Edinburgh, Teknowledge, SAIC, and Northwestern University. The other team is coordinated by SRI and includes research groups from the University of Texas at Austin, ISI/USC, Stanford University, Boeing, the University of Massachusetts at Amherst, Veridian/PSR, and Northwestern University. The component technology developers include Stanford University, MIT, Northwestern University, George Mason University, University of West Florida, and Pragati.

The RKF challenge problems are designed to test SMEs' abilities to use RKF tools to build knowledge bases in two main categories related to understanding biological weapons: textbook knowledge and expert knowledge. The textbook knowledge challenge problem covers standard textbooks in undergraduate biology and is designed to drive the development of the technology that will enable an SME to develop knowledge bases for domains in which knowledge is already relatively well-organized, self-contained or comprehensive, and for which there may exist recognized tests of subject understanding defined independently of the RKF Program. These tests will be based on questions appearing in the textbooks themselves or in associated academic tests for the relevant subject. The expert knowledge challenge problem, on the other hand, covers practical, hands-on tasks for military analysis regarding defensive biological warfare. This problem is designed to drive the development of the technology that will enable the professional SME to develop knowledge that is task-oriented, is not necessarily written down in books, and is germane for an application of national security importance. No appropriate, independently defined test exists now, so it will be necessary to define appropriate tests in the context of the RKF Program.

This paper presents the Disciple technology developed by the George Mason University Learning Agents Laboratory, and its application to the COA challenge problem. This is an important result of the HPKB Program that is representative of the many accomplishments of the program. Disciple is now evolving from a tool for knowledge engineers to a tool that can be used directly by SMEs, as demonstrated by its successful knowledge acquisition experiment performed at the US Army Battle Command Battle Lab. In that sense Disciple is a good illustration of the transition from HPKB to RKF.

This paper presents a learning agent shell and methodology for building knowledge bases and agents, and their innovative application to the development of a critiquing agent for military courses of action, a challenge problem set by DARPA's High Performance Knowledge Bases program. The learning agent shell includes a general problem solving engine and a general learning engine for a generic knowledge base structured into two main components: an ontology that defines the concepts from an application domain, and a set of task reduction rules expressed with these concepts. The development of the critiquing agent was done by importing ontological knowledge from CYC and by teaching the agent how an expert performs the critiquing task. The learning agent shell, the methodology, and the developed critiquer were evaluated in several intensive studies, demonstrating very good results.

A great challenge for Artificial Intelligence is the development of theories, methods and tools that would allow users that do not have knowledge engineering or computer science experience, to build knowledge-bases and agents by themselves. We believe success in this area will have an even greater impact on our society than the development of personal computers. Indeed, if personal computers allowed every person to become a computer user, without the need for special training in computer science, solutions to this AI challenge would allow any such person to become an agent developer. This would lead to a large scale use of computers as personalized intelligent assistants, helping their users in a wide range of tasks. The key issue is that the development of such an

agent should be as easy for the user as it currently is to use a word processor.

In this paper we present recent progress made in the George Mason University Learning Agents Laboratory toward this goal. First we will introduce the concept of a learning agent shell as a tool to be used by directly by an SME to develop an agent. Then we will present the Disciple family of learning agent shells and the successful application of one member of this family, Disciple-COA, to the HPKB's course of action (COA) challenge problem. We will describe the challenge problem and the Disciple-COA shell and methodology used to build the COA critiquing agent. After that we will present the results of the DARPA's evaluation of the developed tools and COA critiquers. We will also briefly present the results of a separate knowledge acquisition experiment with Disciple-COA. We will conclude the paper with a discussion of this research.

Learning Agent Shell

We have introduced the concept of a learning agent shell as a new class of tools for rapid development of practical end-to-end knowledge-based agents, by domain experts, with limited assistance from knowledge engineers (Tecuci et al., 1999). A learning agent shell includes a general problem-solving engine and a general learning engine for building a knowledge base consisting of an ontology and a set of problem solving rules. The process of developing a knowledge-based agent for a specific application relies on importing ontological knowledge from existing knowledge repositories, and on teaching the agent how to perform various tasks, in a way that resembles how an expert would teach a human apprentice when solving problems in cooperation (see Figure 1). This process is based on mixed-initiative reasoning that integrates the complementary knowledge and reasoning styles of the subject matter expert and the agent, and on a division of responsibility for those elements of knowledge engineering for which they have the most aptitude, such that together they form a complete team for knowledge base development.

The concept of learning agent shell is an extension of the concept of expert system shell [Clancey 1984]. As an expert system shell, it includes a general inference engine that can be reused for multiple applications. In addition, a learning agent shell exhibits an organization of the knowledge base into an ontology that specifies the terms from a particular domain, and a set of problem solving rules expressed with these terms.

The ontology is the more general component of the knowledge base, being characteristic to an entire domain, such as medicine, or military. A domain ontology specifies terms that are useful in a wide range of different applications in that domain. For instance, a military ontology would include specifications of military units and of military equipment that are very likely to be included in the knowledge base of any agent developed for a particular military application. Moreover, there is a wide agreement in any mature domain on the basic terms of that domain. This allows one to reuse ontological knowledge that was previously developed, in order to build a new knowledge base. As a consequence, a learning agent shell should include modules for importing ontological knowledge from existing knowledge bases.

The problem solving rules represent the specific component of the knowledge base. The rules are not only specific to a particular application in a given domain, but they are even specific to a particular subject matter expert. Consider, for instance, an agent that assists a military commander in critiquing courses of action with respect to the principles of war and the tenets of army operations (an agent that will be described in more detail in this paper). The rules will identify strengths and weaknesses in a military course of action, and will obviously be domain specific. Moreover, they are very likely to include subjective elements that are

developing a knowledge-based agent relies on importing ontological knowledge…, and on teaching the agent …, in a way that resembles how an expert would teach a human apprentice when solving problems in cooperation

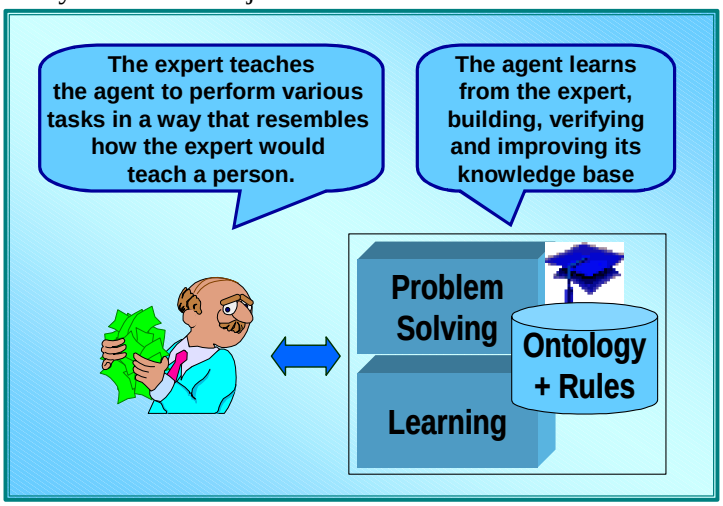


Figure 1: Interacting with a learning agent shell.

based on the experience of a specific military expert. Defining such problem solving rules is a very complex knowledge engineering task. Therefore, the learning agent shell should include mixed-initiative methods for learning such rules from a natural interaction with a subject matter expert.

The Disciple Shell

Over the years we have developed a series of increasingly more advanced learning agent shells from the Disciple family. The problem-solving engine of a Disciple agent is based on the general task reduction paradigm. In this paradigm, a task to be accomplished by the agent is successively reduced to simpler tasks until the initial task is reduced to a set of elementary tasks that can be immediately performed. The ontology of a Disciple agent is based on the frame knowledge model of the Open Knowledge Base Connectivity (OKBC) protocol. OKBC has been developed as a standard for accessing knowledge bases stored in different frame representation systems (Chaudhri et al. 1998). It provides a set of operations for a generic interface to such systems. There is also an ongoing effort of developing OKBC servers for various systems, such as CYC (Lenat, 1995), Ontolingua (Farquhar et al. 1996) and Loom (MacGregor 1991). These servers are becoming repositories of reusable ontologies and domain theories, and can be accessed using the OKBC protocol. The use of the OKBC knowledge model for the Disciple ontology facilitates the import of ontological knowledge from the OKBC compliant knowledge repositories. Because of using the task reduction paradigm, the problem solving rules of a Disciple agent are task reduction rules. Disciple uses an original representation of partially learned rules, called plausible version space. Much of the power of the Disciple approach comes from this plausible version space representation and from the multistrategy learning methods used to learn them. They allow Disciple to incrementally learn a rule, starting from only one example, and to immediately use it in problem solving.

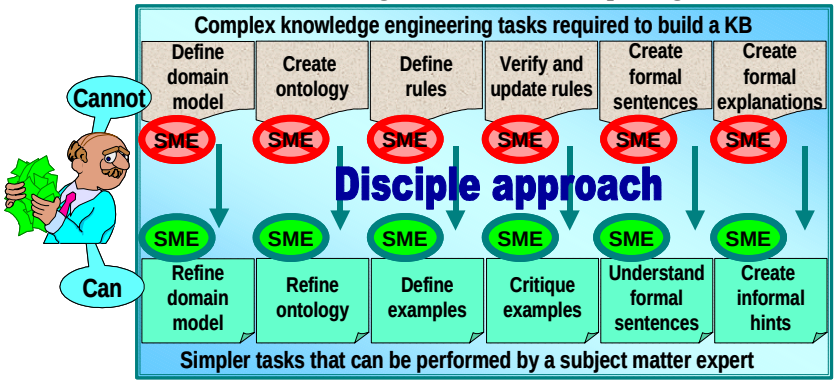


Figure 2: The general strategy behind the Disciple approach.

The Disciple approach is based on several levels of synergism between the expert that has the knowledge to be formalized and the agent that incorporates knowledge engineering methods to formalize it. This multi-level synergism is achieved through mixed-initiative reasoning that integrates complementary human and automated reasoning to take advantage of their respective knowledge, reasoning styles and computational strengths. The mixed-initiative reasoning is based on a division of responsibility between the expert and the agent for those elements of knowledge engineering for which they have the most aptitude, such that together they form a complete team for the development of the agent's knowledge base. From the point of view of the subject matter expert, this mixed-initiative approach results in the replacement of the difficult knowledge engineering tasks required to build a knowledge base, tasks that cannot be performed by a subject matter expert, with simpler tasks that can be performed by the expert, as shown in Figure 2.

To build a knowledge base, we have first to develop a model of the application domain that will make explicit, at a qualitative and informal level, the way the expert solves problems. In the case of Disciple, this means modeling the process of solving a specific problem as a sequence of qualitative and informal problem reduction steps, where a complex problem is successively reduced to simpler problems. Then we have to build an ontology that will define the terms used to express the problems and their solutions. After that, we have to define formal problem solving rules, to verify the rules, and to update them. In general, these processes require the creation and modification of formal sentences and of formal explanations. A subject matter expert cannot perform these tasks. In fact they are very hard even for a knowledge engineer, leading to the well-known knowledge

acquisition bottleneck in the creation of knowledge-based agents.

In the Disciple approach, the difficult knowledge engineering tasks required to develop a knowledge-based agent are replaced with simpler tasks that can be performed by a subject matter expert, with limited assistance from a knowledge engineer. Instead of developing a complete model of the application domain, the expert would need to start with an initial model, extending and refining it with the help of the Disciple agent. The use of the OKBC knowledge model for the Disciple ontology facilitates the import of ontological knowledge from the OKBC compliant knowledge repositories. Therefore, instead of creating an ontology from scratch, the expert would only need to update and extend an imported ontology (Boicu et al. 1999). Instead of defining a complex problem solving rule, the expert would only need to define a specific example of a problem solving episode because Disciple will be able to learn a rule from that example. Instead of debugging a complex problem solving rule, the expert would only need to critique specific examples of problem solving episodes and Disciple will accordingly update the corresponding rule.

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

The COA Challenge Problem

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, selection of a COA, and the preparation of 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 then to 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, can systematically assess selected aspects of a COA, and can suggest repairs to it. The role of this agent is to act as an assistant to the military commander, helping the commander to choose between several COAs under consideration for a certain mission. The agent could also help students to learn to develop courses of action.

The input to the COA critiquing agent consists of the description of a COA that includes the following aspects:

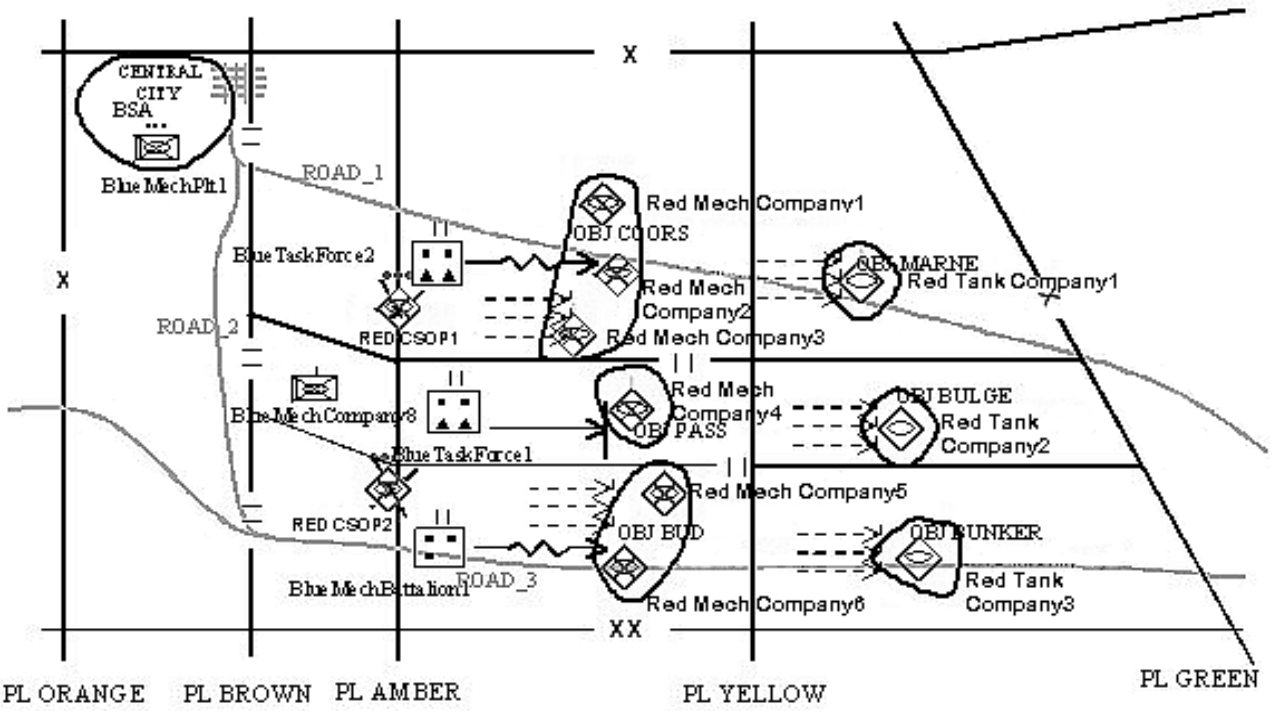
1) The COA sketch, such as the one in the top part of Figure 3, is a graphical depiction of the preliminary plan being considered. It includes enough of the high level structure and maneuver aspects of the plan to show how the actions of each unit fit together to accomplish the overall purpose, while 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.

2) The COA statement, such as the partial one shown in the bottom part of Figure 3, clearly explains what the units in a course of action will do to accomplish the assigned mission. This text includes a description of the mission and the desired end state, as well as standard elements that describe purposes, operations, tasks, forms of maneuver, units, and resources to be used in the COA. The COA statement is expressed in a restricted but expressive subset of English.

3) Selected products of mission analysis, such as the areas of operations of the units, avenues of approach, key terrain, unit combat power, and enemy COAs.

Based on this input, the critiquing agent has to assess various aspects of the COA, such as its viability (its suitability, feasibility, acceptability and completeness), its correctness (which considers the array of

In the Disciple approach, the difficult knowledge engineering tasks required to develop a knowledge-based agent are replaced with simpler tasks that can be performed by a subject matter expert, with limited assistance from a knowledge engineer



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.

Close: BLUE-TASK-FORCE1, a balanced task force (MAIN-EFFORT) 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.

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.

...

Figure 3: A sample of a COA sketch and a fragment of a COA statement.

forces, the scheme of maneuver, and the command and control), and its strengths and weaknesses with respect to the principles of war and the tenets of army operations. The critiquing agent should also be able to clearly justify the assessments made, and to propose improvements to the COA.

General Presentation of Disciple-COA

In the HPKB Program, the COA challenge problem was solved by developing an integrated system composed of four critiquers, each built by a different team, to solve a part of the overall problem: a joint team from Teknowledge and Cycorp, the Expect team from ISI/USC, the Loom team from ISI/USC, and the Disciple team from GMU. All these critiquers shared an input ontology which contains the terms needed to represent the COAs.

The COAs to be critiqued were provided by Alphatech. As presented in the previous section, each such COA is represented by a sketch and a textual description. A statement translator (developed by AIAI of the University of Edinburgh), a COA sketcher (developed by Teknowledge), and a geographic reasoner (developed by Northwestern University) transform and fuse these external representations into a description in the CYC language, according to the input ontology. This description is further used by all the critiquers.

SAIC completed the integrated system with a solution viewer that provided a uniform presentation of the critiques generated by the four developed critiquers.

Our critiquer, called Disciple-COA, 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 some of the strengths of the COA from Figure 3 with respect to the Principle of Mass, identified by Disciple-COA.

In addition to generating answers in natural language, Disciple also provides the reference material based on which the answers are generated, as shown in the bottom part of Figure 4. Also, the Disciple-COA agent can provide justifications for the generated answers at three levels of detail, from a very abstract one that shows the general line of reasoning followed, to a very detailed one that indicates each of the knowledge pieces used in generating the answer.

Figure 5 shows the main modules of Disciple-COA and their interactions. The ontology import module was used to integrate the input COA ontology into Disciple's knowledge base and to translate the CYC representation of specific COAs into Disciple's representation.

After the description of a specific COA, such as the one from Figure 3, has been loaded into Disciple's knowledge base, the subject matter expert may teach the Disciple agent how to critique it. The expert invokes the Cooperative Problem Solver, selects an initial critiquing task (such as "Assess COA with respect to the Principle of Surprise") and asks the Disciple agent to solve it. Disciple uses its task reduction rules to reduce the current task to simpler tasks, showing the expert the reductions found. The expert may accept a reduction proposed by the agent, may reject it or may decide to define himself or herself a new reduction, as illustrated in Figure 5.

To define a new reduction the expert uses the Example Editor. This, in turn, may invoke several knowledge elicitation tools, such as the Object Editor, the Feature Editor or Task Editor, if the specification of the example involves new knowledge elements that are not present in the current knowledge base. Once the reduction has been defined the Rule Learner is invoked to generalize the example to a task reduction rule. The Rule Learner automatically invokes the Explanation Generator that tries to find the explanation of why the reduction indicated

Assess COA411 with respect to the Principle of Mass

There is a major strength in COA411 with respect to mass because BLUE-TASK-FORCE1 is the MAIN-EFFORT1 and it acts on the decisive point of the COA (RED-MECH-COMPANY4) with a force ratio of 10.6, which exceeds a recommended force ratio of 3.0. Additionally, the main effort is assisted by supporting action SUPPRESS-MILITARY-TASK1 which also acts on the decisive point. This is good evidence of the allocation of significantly more than minimum combat power required at the decisive point and is indicative of the proper application of the principle of mass.

There is a strength in COA411 with respect to mass because BLUE-TASK-FORCE1 is the main effort of the COA and it has been allocated 33% of available combat power but this is considered just a medium level weighting of the main effort.

There is a strength in COA411 with respect to mass because BLUE-MECH-COMPANY8 is a COMPANY-UNIT-DESIGNATION level maneuver unit assigned to be the reserve. This is considered a strong reserve for a BRIGADE-UNIT-DESIGNATION level COA and would be available to continue the operation or exploit success.

Reference: FM 100-5 pg 2-4, KF 113.1, KF 113.2, KF 113.3, KF 113.4, KF 113.5 - To mass is to synchronize the effects of all elements of combat power at the proper point and time to achieve decisive results. Observance of the Principle of Mass may be evidenced by allocation to the main effort of significantly greater combat power than the minimum required throughout its mission, accounting for expected losses. Mass is evidenced by the allocation of significantly more than minimum combat power required at the decisive point.

Figure 4: Solutions generated by Disciple-COA.

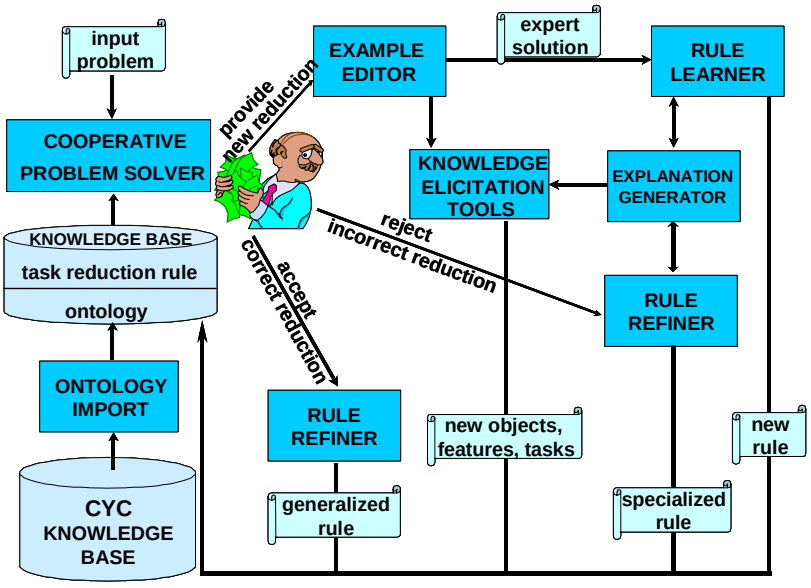


Figure 5: Expert-agent interactions.

by the expert is correct. The Explanation Generator proposes several plausible explanations from which the expert has to select the correct ones. The expert may help the agent to find the correct explanations by providing a hint.

If the expert accepts a reduction proposed by the agent then the Rule Refiner is invoked and may generalize the rule that has led to this reduction.

If the expert rejects a reduction proposed by the agent then the agent attempts to find an explanation of why the reduction is not correct, the Explanation Generator being invoked again, as described above. The explanation found is used by the Rule Refiner to specialize the rule.

After a new rule is learned or an existing rule is refined, the Cooperative Problem Solver resumes the task reduction process until a solution of the initial problem is found.

In addition to the Cooperative Problem Solver, Disciple-COA also includes an autonomous problem solver that is used after Disciple-COA has been trained.

The next sections describe in more detail this process of developing the Disciple-COA agent.

Ontology Development

For Disciple-COA, an initial ontology was defined by importing the input ontology built by Teknowledge and Cycorp for the COA challenge problem. The imported ontology was further developed by using the ontology building tools of Disciple. These tools include specialized browsers and editors for the various knowledge pieces of Disciple (e.g. the object editor, the object feature editor, the task feature editor, the hierarchy browser and the association browser).

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 a

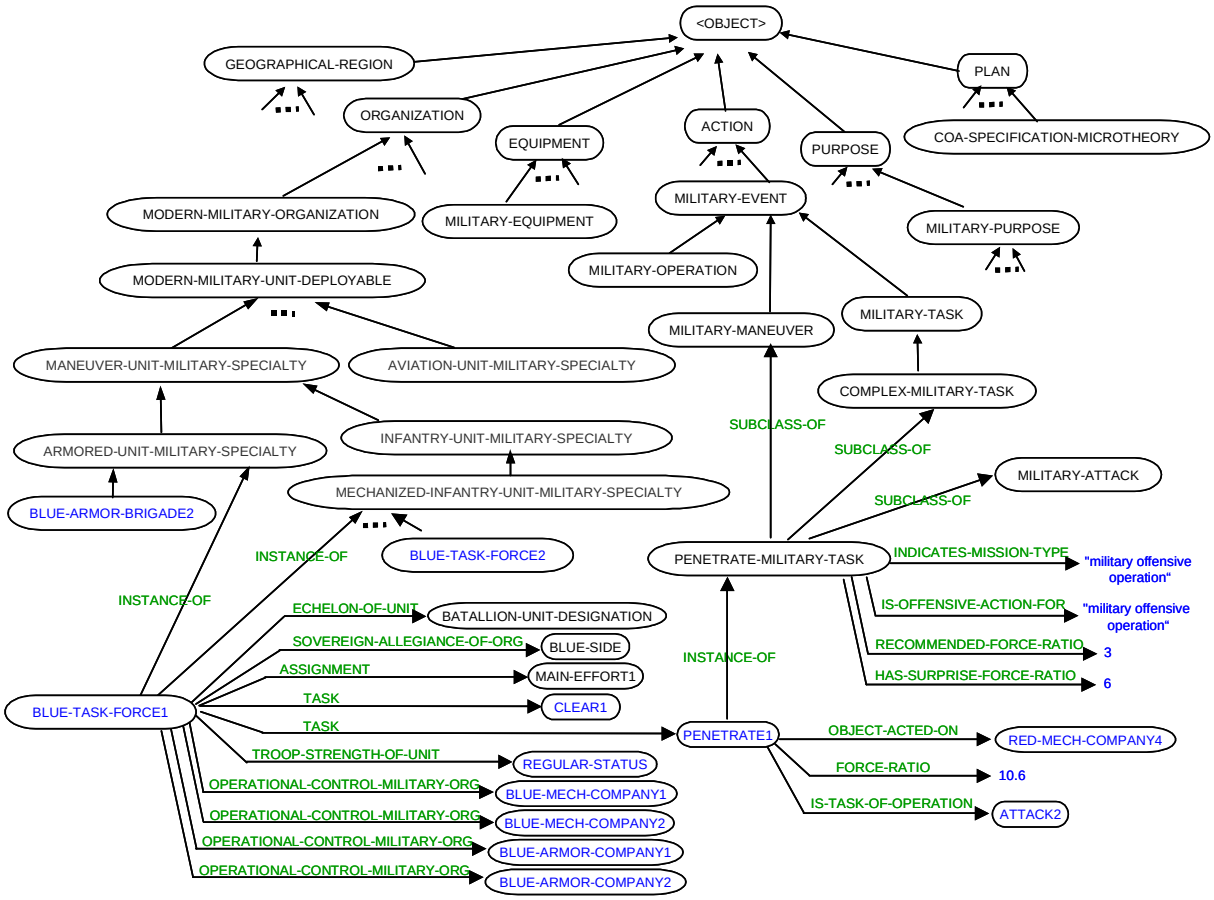


Figure 6: Fragment of the ontology.

fragment of the object ontology. The top part represents the upper level of the object ontology that identifies the types of concepts represented in the ontology. They include GEOGRAPHICAL-REGION, ORGANIZATION, EQUIPMENT and ACTION. Each of these concepts is the top of a specialized hierarchy, such as the hierarchy of organizations showed in the left part of Figure 6. 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 bottom part 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 at the battalion level, 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 is PENETRATE1. 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.

Training the Disciple-COA agent

The next step in the development of the Disciple-COA critiquer was to teach Disciple to critique COAs with respect to the principles of war and the tenets of army operations. The expert loads the description of a specific COA, such as COA411 represented in Figure 3, and then invokes the Cooperative Problem Solver with a task of critiquing the COA with respect to a certain principle or tenet. Disciple uses its task reduction rules to reduce the current task to simpler tasks, showing the expert the reductions found. The expert may accept a reduction proposed by the agent, may reject it or may decide to define a new reduction. From each such interaction Disciple will either refine a previously learned rule or will learn a new task reduction rule. After a new rule is learned or an existing rule is refined, the Cooperative Problem Solver resumes the task reduction process until a solution of the initial problem is found.

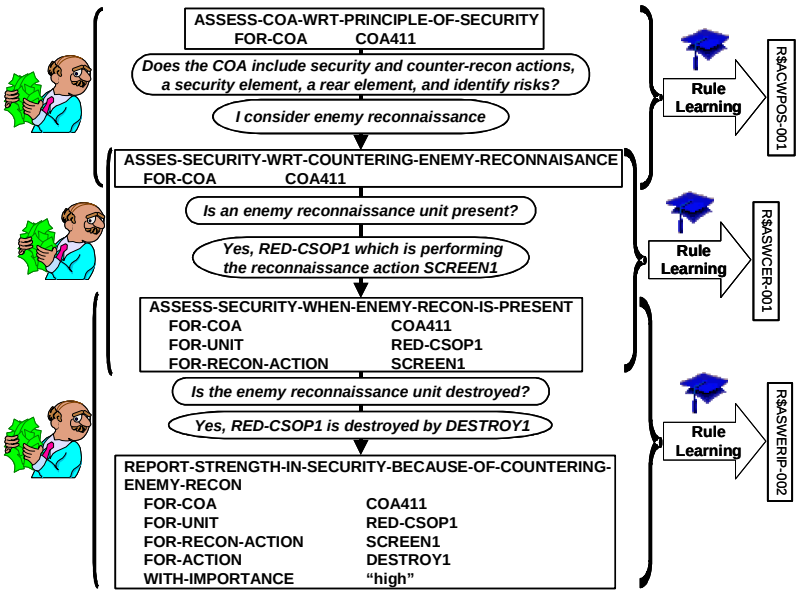


Figure 7: Task reductions indicated by the expert.

Initially Disciple does not contain any rules. Therefore, all the problem solving steps (i.e. task reductions) must be provided by the expert, as illustrated in Figure 7, and explained in the following.

To assess COA411 with respect to the Principle of Security the expert and Disciple need a certain amount of information which is obtained by asking a series of questions. The answer to each question allows one to reduce the current assessment task to a more detailed one. This process continues until the expert and Disciple have enough information about COA411 to make the assessment. As shown in Figure 7, the initial task is reduced to that of assessing the security of COA411 with respect to the countering of enemy reconnaissance. Then one 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 for COA411 with respect to countering enemy reconnaissance is now reduced to the better defined task of assessing security when 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 countered.

Figure 8 illustrates the process of teaching Disciple. The left hand side represents the reasoning process of the expert, the question and the answer being in free natural language format. While this line of reasoning is very natural to a human expert, a learning agent cannot understand it. The explanation that would be understood by the agent is represented in the upper right part of Figure 8, and consists of various relations between certain elements from its ontology. The first explanation piece states, in Disciple's formal language, that RED-CSOP1 is an enemy unit. The second explanation piece expresses the fact that RED-CSOP1 is performing the action SCREEN1. Finally, the last explanation piece expresses the fact that SCREEN1 is a reconnaissance action. While an expert can understand the meaning of these formal expressions, he cannot define them because he is not a knowledge engineer. For one thing, he would need to use the formal language of the agent. But this would not be enough. He would also need to know the names of the potentially many thousands of concepts and features from the agent's ontology.

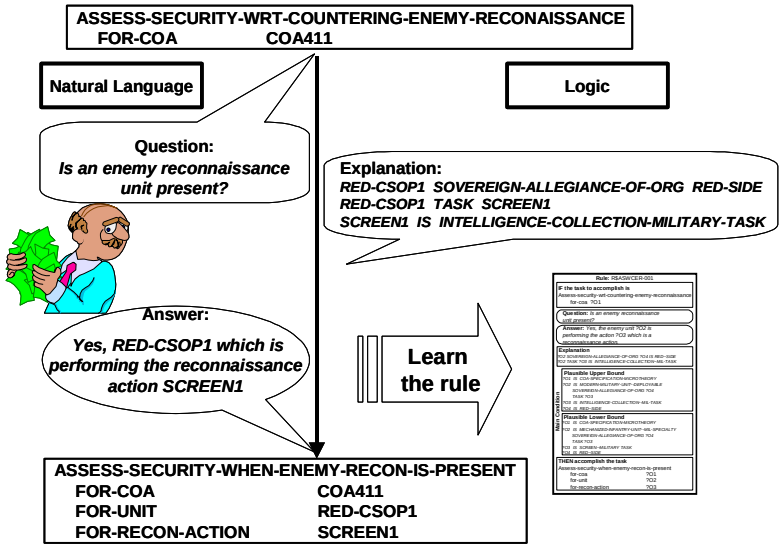


Figure 8: Teaching Disciple to reduce a task.

While defining the formal explanations of this task reduction step is beyond the individual capabilities of the expert and the agent, it is not beyond their joint capabilities. Finding these explanation pieces is a mixed-initiative process of searching the agent's ontology, an explanation piece being a path of objects and relations in this ontology. In essence, the agent will use analogical reasoning and help from the expert to identify and propose a set of plausible explanation pieces from which the expert will have to select the correct ones. One explanation generation strategy is based on an ordered set of heuristics for analogical reasoning. These heuristics exploit the hierarchies of objects, features and tasks to identify the rules that are similar to the current reduction, and to use their explanations as a guide to search for similar explanations of the current example. This cooperative explanation-generation process proved to be very effective, as demonstrated by the successful knowledge acquisition experiment described in this paper.

From the example reduction and its explanation in Figure 8, Disciple automatically generated the plausible version space rule in Figure 9. This is an IF-THEN rule, the components of which are generalizations of the elements of the example in Figure 8. In addition, the rule contains two conditions for its applicability, a plausible lower bound condition and a plausible upper bound condition. These conditions approximate an exact applicability condition that Disciple attempts to learn. Initially, the plausible lower bound condition covers only the example in Figure 8, restricting the variables from the rule to take only the values from this example. It also includes the relations between these 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. The domain of a feature is the set

of objects that may have that feature. The range is the set of possible values of that feature. 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".

The learned PVS rules, such as the one in Figure 9, are used in problem solving to generate task reductions with different degrees of plausibility, depending on which of its conditions are satisfied. If the Plausible Lower Bound Condition is satisfied, then the reduction is very likely to be correct. If the Plausible Lower Bound Condition is not satisfied, but the Plausible Upper Bound Condition is satisfied, then the solution is considered only plausible. Any application of a PVS rule however, either successful or not, provides an additional (positive or negative) example, and possibly an additional explanation, that are used by the agent to further improve the rule through the generalization and/or specialization of its conditions.

Let us consider again the specific task reductions from Figure 7. At least for the elementary tasks, such as the one from the bottom of the figure, the expert needs also to express them in natural language:

"There is a strength with respect to surprise in COA411 because it contains aggressive security/counter-reconnaissance plans, destroying enemy intelligence collection units and activities. Intelligence collection by RED-CSOP1 will be disrupted by its destruction by DESTROY1".

Similarly, the expert would need to indicate the source material for the concluded assessment. The learned rules will contain generalizations of these phrases that are used to generate answers in natural language, as illustrated in Figure 4. Similarly, the generalizations of the questions and the answers from the rules applied to generate a solution are used to produce an abstract justification of the reasoning process.

As Disciple-COA learns plausible version space rules, it can use them to propose routine or innovative solutions to the current problems. The *routine solutions* are those that satisfy the plausible lower bound conditions of the rules and are very likely to be correct. Those that are not correct are kept as exceptions to the rule. The *innovative solutions* are those that do not satisfy the plausible lower bound conditions but satisfy the plausible upper bound conditions. These solutions may or may not be correct, but in each case will lead to a refinement of the rules that generated them. Let us consider the situation illustrated in Figure 10. After it has been shown how to critique COA411 with respect to the principle of security, Disciple is asked to critique COA421. COA421 is similar with COA411 except that, in this case the enemy recon unit is not destroyed. Because of this similarity, Disciple-COA is able to propose the two top reductions in Figure 4. Both of them are innovative reductions that are accepted by the expert. Therefore Disciple-COA generalizes the plausible lower bound conditions of the corresponding rules, as little as possible, so as to cover these reductions and to remain less general or at

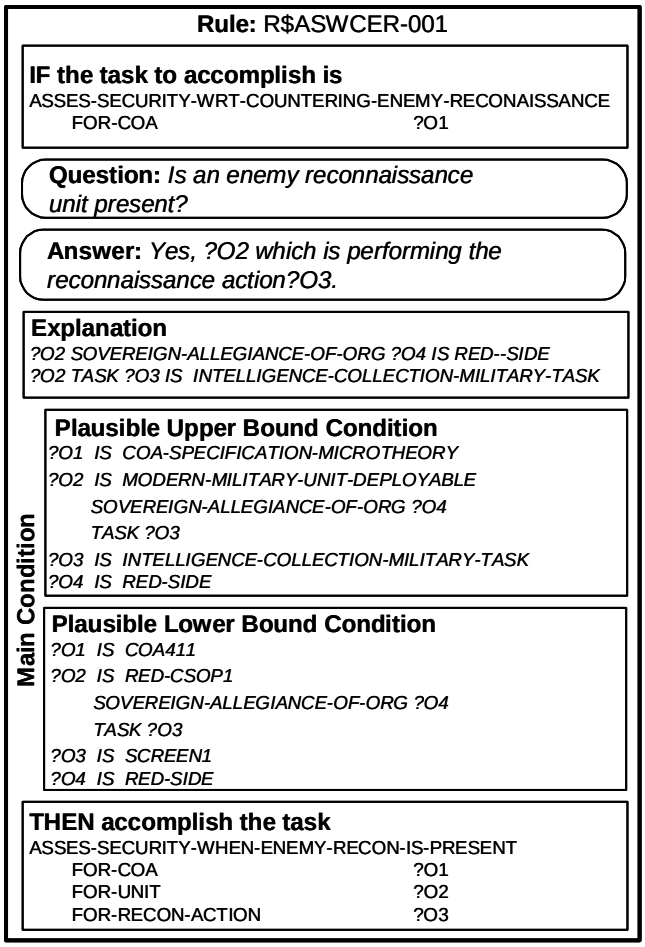


Figure 9: Plausible version space rule learned from the example and the explanation in Figure 8.

As Disciple learns plausible version space rules, it can use them to propose routine or innovative solutions to the current problems.

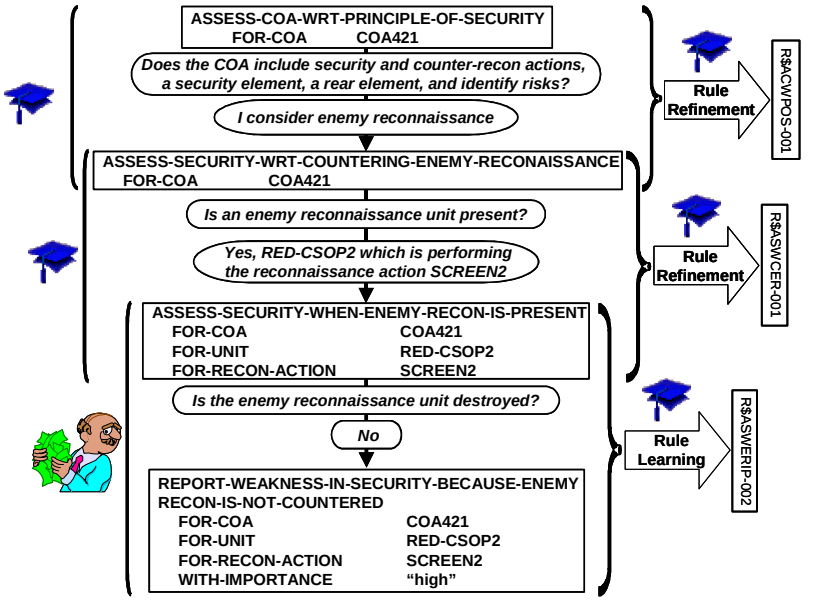


Figure 10: Cooperative problem solving and learning.

most as general as the corresponding plausible lower bound conditions.

The last reduction step in Figure 10 has to be indicated by the expert because no rule of Disciple-COA is applicable. We call the expert-provided reduction a creative problem solving step. From each such reduction Disciple-COA will learn a new task reduction rule, as was illustrated above.

Through refinement, the task reduction rules may become significantly more complex than the rule in Figure 9. For instance, when a reduction proposed by Disciple is rejected by the expert, Disciple will attempt to find an explanation of why the reduction is wrong. Then the rule may be refined with an Except-when plausible version space condition. The bounds of this version space are generalizations of the explanations that should not hold in order for the reduction rule to be applicable.

In any case, comparing the left hand side of Figure 8 (which is defined by the domain expert) with the rule from Figure 9 (which is learned by Disciple) suggests the usefulness of Disciple for knowledge acquisition. In the traditional knowledge engineering approach, a knowledge engineer would need to manually define and debug a rule like the one in Figure 9. With Disciple, the domain expert needs only to define an example reduction, because Disciple will learn and refine the corresponding rule. That this approach works very well is demonstrated by the intense experimental studies conducted with Disciple and reported in the next section.

Evaluation of the COA Critiquers

In addition to GMU, other three research groups have developed COA critiquers as part of the HPKB program. Teknowledge and CYC have developed a critiquer based on the CYC system (Lenat, 1995). The other two critiquers have been developed at ISI, one based on the Expect system (Kim and Gil, 1999), and the other based on the Loom system (MacGregor, 1999). All the critiquers were evaluated as part of the HPKB's annual evaluation that took place during the period July 6-16, 1999, and included five evaluation items of increasing difficulty. Each item consisted of descriptions of various COAs and a set of questions to be answered about each of them. Item1 consisted of COAs and questions that were previously provided by DARPA to guide the development of the COA critiquing agents. Item2 included new test questions about the same COAs. Items 3, 4, and 5 consisted of new COAs that were increasingly more complex and required further development of the COA agents in order to properly answer the asked questions. Each of the Items 3, 4 and 5 consisted of two phases. In the first phase each team had to provide initial system responses. Then the evaluator issued the model answers and each team had a limited amount of time to repair its system, to perform further knowledge acquisition, and to generate revised system responses.

The responses of each system were scored by a team of domain experts along the following dimensions and associated weights: Correctness-50% (matches model answer or is otherwise judged to be correct), Justification-30% (scored on presence, soundness, and level of detail), Lay Intelligibility-10% (degree to which a lay observer can understand the answer and the justification), Sources-10% (degree to which appropriate sources are noted), and Proactivity-10% extra credit (appropriate corrective actions or other information suggested to address the critique). Based on these scores several classes of metrics have been computed, including Recall and Precision. Recall is obtained by dividing the score for all answers provided by a critiquer to the total number of model answers for the asked questions. This was over 100% in the case of our critiquer, primarily because of

the extra credit received for generating additional critiques that were not among the model answers provided by the evaluator. "Precision" is obtained by dividing the same score by the total number of answers provided by that system (both the model answers provided by the evaluator and the new answers provided by the critiquer). The results obtained by the four evaluated critiquers are presented in Figure 11. These graphs show also the averages of these results which represent the recall and the precision of the integrated system.

Figure 12 compares the recall and the coverage of the developed critiquers for the last three most complex items of the evaluation. For each item, the beginning of each arrow shows the coverage and recall for the initial testing phase, and the end of the arrow shows the same data for the modification phase. In this graph, the results that are above and to the right are superior to the other results. This graph also shows that all the systems increased their coverage during the evaluation. In particular, the KB of Disciple increased by 46% (from the equivalent of 6229 simple axioms to 9092 simple axioms), which represents a very high rate of knowledge acquisition of 286 simple axioms/day.

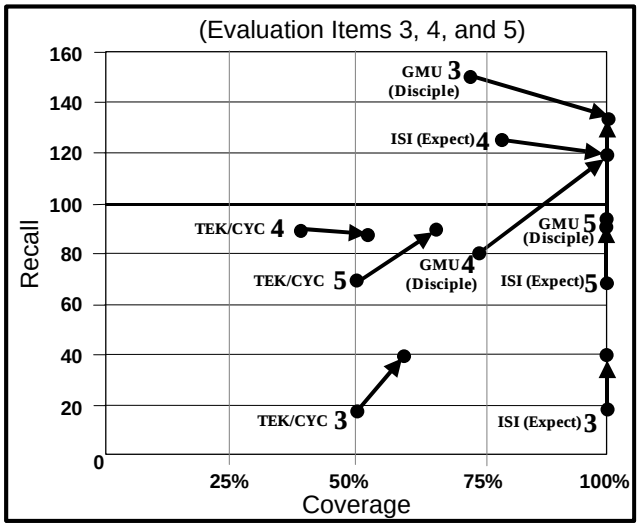


Figure 12: Coverage vs Recall, Pre- and Post-Repair

Direct Knowledge Acquisition from SMEs

During August 1999 we conducted a one week knowledge acquisition experiment with Disciple-COA, at the US Army Battle Command Battle Lab, in Fort Leavenworth,

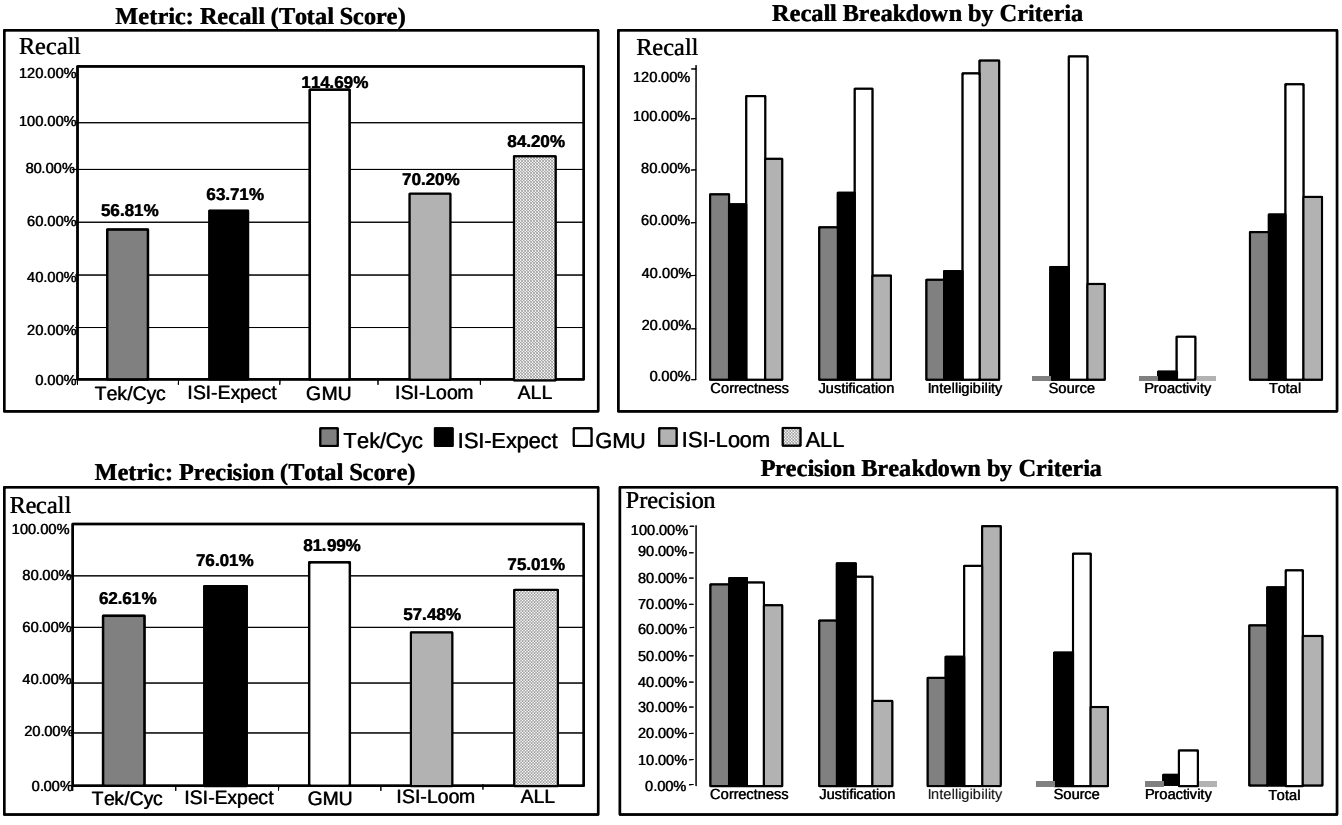


Figure 11: The performance of the developed COA critiquers and of the integrated system.

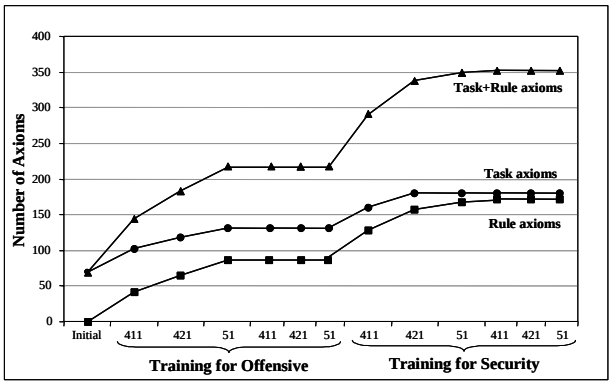


Figure 13. The evolution of the knowledge base during the knowledge acquisition experiment.

Kansas, to test the claim that domain experts that do not have prior knowledge engineering experience can teach Disciple-COA. The experiment involved four such military experts and had three phases: a joint training phase (day 1 to 3), an individual teaching experiment (day 4), and a joint discussion of the experiment (day 5). The entire experiment was videotaped. The training for the experiment included a detailed presentation of Disciple's knowledge representation, problem solving and learning methods and tools. For the teaching experiment, each 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. We also provided the experts with the descriptions of three COAs, COA411, COA421, and COA51, to be used for training Disciple. These were the COAs used in the final phases of the DARPA's evaluation of all the critiquers. Finally, we provided and discussed with the experts the modeling of critiquing these COAs with respect to the principles of offensive and security. That is, we provided the experts with specific task reductions like the one from Figure 7, to guide them in teaching Disciple-COA. After that, each expert taught Disciple-COA independently, while being supervised by a knowledge engineer whose role was to help the expert if he reached an impasse while using Disciple.

Figure 13 shows the evolution of the knowledge base during the teaching process for one of the experts, being representative for all the four experts. In the morning the expert taught Disciple to critique COAs with respect to the Principle of Offensive and in the afternoon he taught it to critique COAs with respect to the Principle of Security. In both cases the expert used first COA411, then COA422 and then COA51. As one can see from Figure 13, Disciple initially learned more rules, and then the emphasis shifted on rule refinement. Therefore, the increase in the size of the knowledge base 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 took place in the afternoon and consisted of 72 minutes of expert-Disciple interactions. During this time Disciple learned 14 tasks and 12 rules (136 simple axioms equivalent). There was no or 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, or 98 simple axioms equivalent/hour). At the end of this training process, Disciple-COA was able to correctly identify 17 strengths and weaknesses of the 3 COAs with respect to the principles of offensive and security.

After the experiment, each expert was asked to fill in a detailed questionnaire designed to collect subjective data for usability evaluation. All the answers took into account that Disciple-COA was a research prototype and not a commercial product, and were rated based on a scale of agreement with the question from 1 to 5, with 1 denoting not at all and 5 denoting very. For illustration, Table 1 shows three questions and the answers provided by the four experts.

Conclusions

We have introduced the concept of learning agent shell and a methodology for rapid development of knowledge bases and agents based on the Disciple learnig agent shell. The Disciple shell and methodology have been applied to the development of a critiquing agent that acts as an assistant to a military commander. This approach and the developed agent have been evaluated in two intensive studies. The first study concentrated on the quality of the developed critiquer and the ability to rapidly extend it by its developers and subject matter experts. The second study concentrated on the ability of domain experts to extend the knowledge

base of the critiquer with very limited assistance from knowledge engineers. Both studies have shown that Disciple has reached a significant level of maturity, being usable to rapidly develop complex knowledge based agents.

The two main factors that contributed to the success of Disciple-COA are: 1) the synergistic collaboration between the subject matter expert and Disciple in developing the knowledge base, and 2) the multistrategy learning method of Disciple which is based on the plausible version space representation. Our research on plausible version spaces has its origins in Mitchell's influential work on version spaces and his candidate elimination algorithm (Mitchell, 1997), extending them along several dimensions, and leading to a powerful and practical mixed-initiative multistrategy learning approach that synergistically integrates a wide range of knowledge acquisition and machine learning strategies, including apprenticeship learning, empirical inductive learning from examples and explanations, and analogical learning. This method is based on a powerful knowledge representation language that includes the frame-based OKBC knowledge model for the representation of the ontological knowledge, and complex task reduction rules with multiple conditions. Moreover, we do not make the assumption that the representation space for learning needs to be completely defined before learning can take place. On the contrary, the representation language is assumed to be incomplete and partially incorrect, and is itself evolving during rule learning through the improvement of the ontology. Because the learning process takes place in an evolving representation language, the various plausible bounds of a rule are subject to heuristic transformations that involve both generalization and specialization operations for each bound. These transformations are guided by hints, explanations and analogies. Therefore, the learning process is very efficient and does not suffer from any combinatorial explosion. Also, learning may take place even in the presence of exceptions, when there is no rule that discriminates between the positive examples and the negative examples.

To conclude, our long term vision for Disciple, that guides our future work, is to evolve it to a point where it will allow normal computer users to build and maintain knowledge bases and knowledge based agents, as easily as they now use personal computers for text processing.

Table 1. Sample questions answered by the experts.

Questions	Answers
Do you think that Disciple is a useful tool for Knowledge Acquisition?	• 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?	• Rating 5. Yes. • 5 (absolutely) • 4 • Yes. As it develops and becomes tailored to the user, it will simplify the tedious tasks.
Were the procedures/ processes used in Disciple compatible with Army doctrine and/or decision making processes?	• Rating 5. As a minimum yes, as a maximum—better! • This again was done very well. • 4 • 4

Acknowledgments.

This research was done in the GMU Learning Agents Laboratory (LALAB). Research of the LALAB is sponsored by the Defense Advanced Research Projects Agency (DARPA), the Air Force Office of Scientific Research (AFOSR) and Air Force Research Laboratory, Air Force Material Command, USAF, under agreement number F30602-00-2-0546, grant no. F49620-97-1-0188 and grant no. F49620-00-1-0072. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, AFOSR, the Air Force Research Laboratory, or the U.S. Government. 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. Florin Ciucu, Cristian Levcovici, Cristina Cascaval, Ping Shyr, and Kathryn Wright contributed to Disciple-COA.

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 *Proceedings of the Sixteenth National*

Conference on Artificial Intelligence, 900-901, Menlo Park, California: AAAI Press.

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

Burke, M., *Rapid Knowledge Formation (RKF) Program Description*, http://dtsn.darpa.mil/iso/programtemp.asp?mode=331

Burke, M., *High Performance Knowledge Bases (HPKB) Program Description*, http://dtsn.darpa.mil/iso/index2.asp?mode=9

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

Clancey, W. J. 1984. NEOMYCIN: Reconfiguring a rule-based system with application to teaching. In Clancey W. J. and Shortliffe, E. H., eds. *Readings in Medical Artificial Intelligence*, pp.361-381. Reading, MA: Addison-Wesley.

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.

FM-105. 1993. US Army Field Manual 100-5, Operations, Headquarters, Department of the Army.

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, Alphatech, Inc., Burlington, MA.

Kim, J., and Gil, Y. 1999. Deriving Expectations to Guide Knowledge Base Creation. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 235-241, Menlo Park, California: 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 at: http://www.isi.edu/isd/LOOM/papers/macgregor/Loom_Retrospective.html

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

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

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

Tecuci G., Boicu M., Bowman M., Marcu D., Shyr P., and Cascaval C. 2000 "An Experiment in Agent Teaching by Subject Matter Experts," *International Journal of Human-Computer Studies* 53: 583-610.