In Proceedings of the Seventeenth National Conference on Artificial Intelligence and the Twelfth Conference on Innovative Application of Artificial Intelligence, Texas, July 2000, Menlo Park, CA: AAAI Press, 2000.

Rapid Development of a High Performance Knowledge Base for Course of Action Critiquing

Gheorghe Tecuci (tel: 703 993-1722, email: tecuci@gmu.edu, fax: 703 993-1710)
Mihai Boicu (tel: 703 993-4669, email: mboicu@cs.gmu.edu)
Dorin Marcu (tel: 703 993-4669, email: dmarcu@cs.gmu.edu)
Michael Bowman (tel: 703 704-0529, email: mbowman3@osf1.gmu.edu)
Florin Ciucu (tel: 703 993-1535, email: fciucu@cs.gmu.edu)
Cristian Levcovici (tel: 703 993-1535, email: clevcovi@cs.gmu.edu)

Learning Agents Laboratory, Department of Computer Science, MS 4A5
George Mason University, 4400 University Drive, Fairfax, VA 22030-4444
http://lalab.gmu.edu

Application domain: Critiquing of Military Courses Of Action

AI technique employed and issue addressed:

- Knowledge Acquisition Bottleneck
- Development and implementation of a learning-based methodology and agent shell, called Disciple-COA, for direct knowledge acquisition from domain experts with limited assistance from knowledge engineers.

Tools used

- LISP and JAVA were used to develop the Disciple-COA shell
- Disciple-COA was used to develop the knowledge base for COA critiquing

Application Status

Disciple-COA and its KB for COA critiquing are research prototypes that have been evaluated in several studies as part of DARPA's High Performance Knowledge Bases program.

Abstract

This paper presents a practical learning-based methodology and agent shell for building knowledge bases and knowledge-based 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 agent shell consists of an integrated set of knowledge acquisition, learning and problem solving modules for a generic knowledge base structured into two main components: an ontology that defines the concepts from a specific application domain, and a set of task reduction rules expressed with these concepts. The rapid development of the COA critiquing agent was done by importing an initial ontology from CYC and by teaching the agent to perform its tasks in a way that resembles how an expert would teach a human apprentice when solving problems in cooperation. The methodology, the agent shell, and the developed critiquer were evaluated in several intensive studies, and demonstrated very good results.

1 Introduction

The purpose of this paper is twofold: 1) to present a maturing learning-based methodology and tool for developing knowledge-based agents, and 2) to present an innovative application of this methodology and tool.

This work was performed as part of the High Performance Knowledge Bases (HPKB) program which ran from 1997 to 1999, with support from DARPA and AFOSR (Cohen et al. 1998). The goal of HPKB was to produce the technology needed to rapidly construct large knowledge-bases (with many thousands of axioms) 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 organizations participating in HPKB were given the challenge of solving a selection of knowledge-based problems in a particular domain, and then modifying their systems quickly to solve further problems in the same domain. The aim of the exercise was to test the claim that, with the latest AI technology, large knowledge bases can be built quickly and efficiently.

The George Mason University Learning Agents Lab's approach to HPKB is based on the Disciple apprenticeship multistrategy learning theory, methodology and shell for rapid development of knowledge bases and knowledge-based agents (Tecuci 1998). Stimulated by the HPKB challenge problems, Disciple has been significantly scaled-up. The challenge problem for the first year of HPKB was to build a knowledge-based workaround agent that is able to plan how a convoy of military vehicles can "work around" (i.e. circumvent or overcome) obstacles in their path, such as damaged bridges or minefields (Tecuci et al. 1999). The challenge problem for the second year of HPKB was to build a critiquing agent that can evaluate military Courses of Action (COA) that were developed as hasty candidate plans for ground combat operations. The developed Disciple agents and the Disciple shell were evaluated during intense DARPA annual evaluation periods, together with the other tools and systems developed in the HPKB program by the other participating teams. In both cases the Disciple agents were developed very rapidly and demonstrated performance superior to the other developed systems.

In this paper we will present the successful application of the Disciple approach to the COA challenge problem. We will first describe this challenge problem which in itself represents an innovative application of Artificial Intelligence. Then we will present the Disciple tool 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. We will conclude the paper with a discussion of these results and the future direction of our work.

2 The Course Of Action Challenge Problem

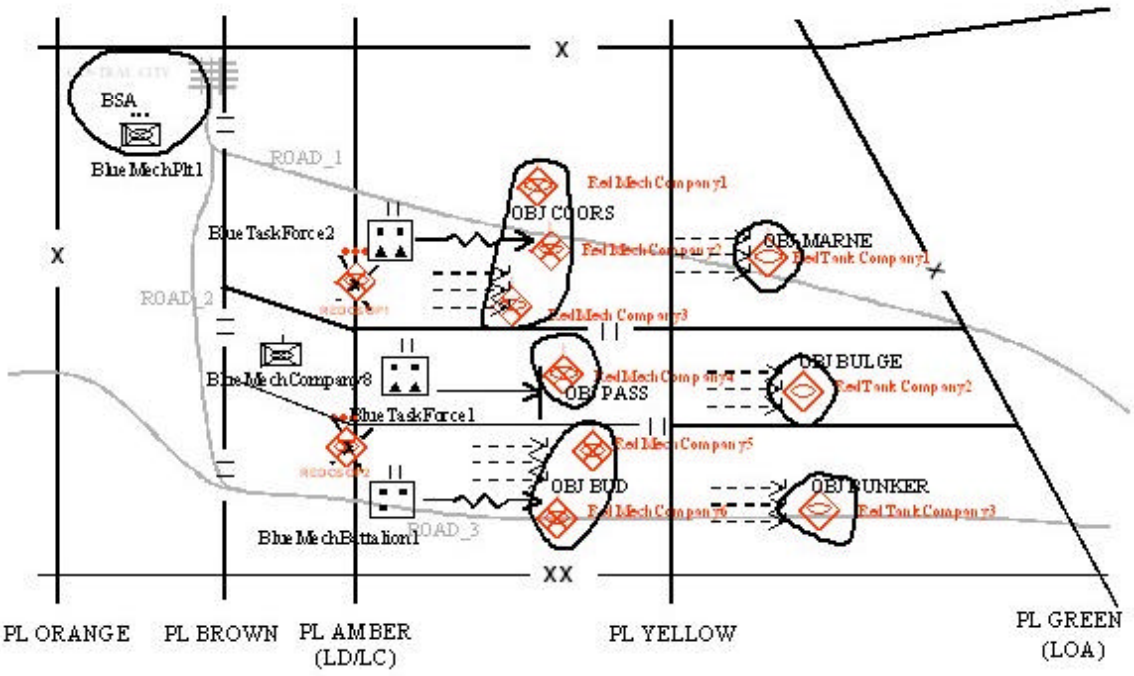
A military COA is a preliminary outline of a plan for how a military unit might attempt to accomplish a mission. A COA is not a complete plan in that it leaves out many details of the operation such as exact initial locations of friendly and enemy forces. After receiving orders to plan for a mission, 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 (Alphatech, 1999).

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

a) The COA sketch, such as the one in the top part of Figure 1, 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



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 1: A sample of a COA sketch and a fragment of a COA statement.

actions and tasks assigned to friendly units. The COA sketch is drawn using a palette-based sketching utility.

b) The COA statement, such as the partial one shown in the bottom part of Figure 1, clearly explains what the units in a course of action will do to accomplish the assigned mission. The text of a COA statement includes a description of the mission and 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.

c) 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 (suitability, feasibility, acceptability and completeness), its correctness (array of forces, scheme of maneuver, command and control), and its strengths and weaknesses with respect to the Principles of War and the Tenets of Army Operations, to justify the assessments made and to propose improvements to the COA.

In the HPKB program, the COA challenge problem was solved by developing an integrated system composed of several critiquers, each built by a different team, to solve a part of the overall problem. The participating teams were Teknowledge-Cycorp-AIAI, ISI/Expect, ISI/Loom, U.Mass, Northwestern Univ. and GMU. All the teams shared an input ontology and used the same internal representation of the input generated by Teknowledge and AIAI from COA descriptions provided by Alphatech.

We developed a COA critiquer, called Disciple-COA, that 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,

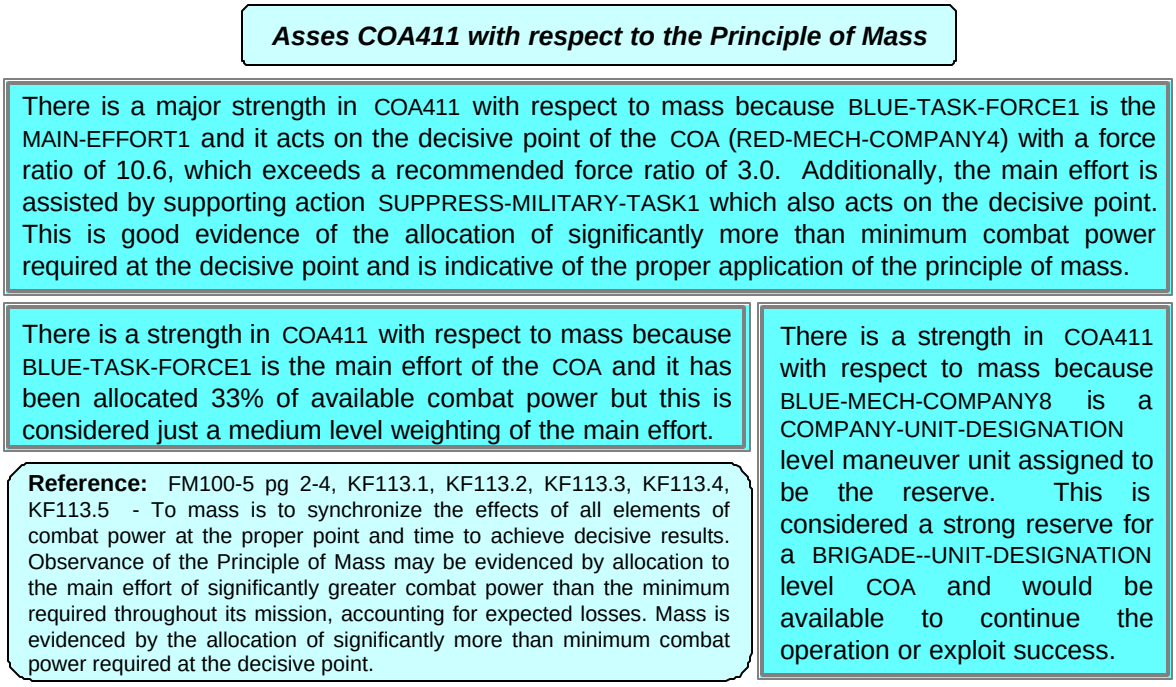


Figure 2: Answers generated by Disciple-COA.

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 2, for instance, shows some of the strengths of the COA from Figure 1 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 left of Figure 2. Also, the Disciple-COA agent can provide justifications of 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.

In the next section we will present the general methodology used to build Disciple-COA and the architecture of Disciple-COA.

3 General presentation of Disciple-COA

Disciple is the name of an evolving theory, methodology and shell for rapid development of knowledge bases and knowledge-based agents, by subject matter experts, with limited assistance from knowledge engineers (Tecuci, 1998). The current Disciple shell consists of an integrated set of knowledge acquisition, learning and problem solving modules for a generic knowledge base structured into two main components: an ontology that defines the concepts from a specific application domain, and a set of problem solving rules expressed with these concepts. The problem solving approach of an agent built with Disciple is task reduction, where 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. Therefore, the rules from the KB are task reduction rules. The ontology consists of hierarchical descriptions of objects, features and tasks, represented as frames, according to the knowledge model of the Open Knowledge Base Connectivity (OKBC) protocol (Chaudhri et al. 1998).

The development of a specific Disciple agent includes the following processes: 1) the customization of the problem solver and the interfaces of the Disciple shell for that particular domain; 2) the building of the domain ontology by importing knowledge from external repositories of knowledge and by manually defining the other components of the ontology, and 3) teaching the agent to perform its tasks, teaching that resembles how an expert would teach a human apprentice when solving problems in cooperation. Following this process we have developed Disciple-COA which is presented in Figure 3.

For Disciple-COA, an initial ontology was defined by importing the input ontology built by Teknowledge and AIAI for the COA challenge problem. The input ontology contains the terms needed to represent the COAs to be critiqued, and was shared by all the developed critiquers. The

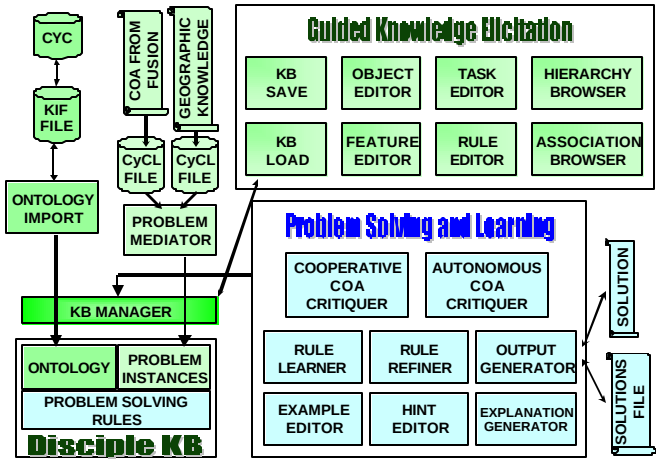


Figure 3: The architecture of Disciple-COA.

top level of this ontology is represented in Figure 4. It includes concepts for representing geographical information, military organizations and equipment, descriptions of specific COAs, military tasks, operations and purposes. As shown in the top left of Figure 3, this ontology was first translated from CYC's language into KIF (Genesereth and Fikes, 1992) and from there it was translated into the representation language of Disciple and the other critiquers.

The imported ontology was further developed by using the ontology building tools of Disciple shown in the top right side of Figure 3 (the object, feature, task and rule editors and browsers).

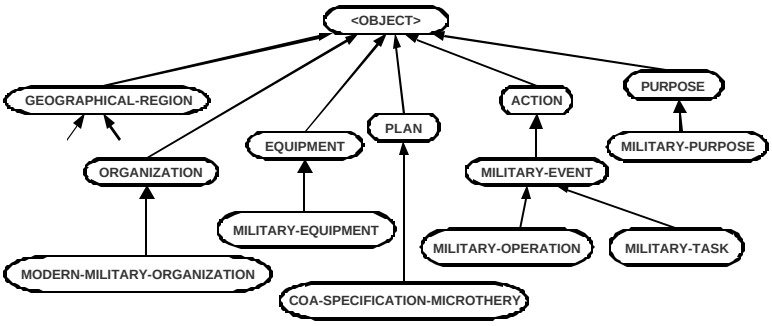


Figure 4: Top level of the ontology imported from CYC.

As presented in the previous section, the COA to be critiqued is represented as a sketch and a textual description. A statement translator (developed by AIAI and Teknowledge), a COA sketcher (developed by Teknowledge), and a geographic reasoner (developed by Northwestern Univ.) transform and fuse these external representations into a description in the CYC language, according to the input ontology. This description is imported into Disciple's ontology by the problem mediator module of Disciple.

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 1, and then invokes the Cooperative Problem Solver with an initial 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 learn a new task reduction rule or will refine a previously learned rule, as explained in the following. 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.

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 5, and explained in the following.

To assess COA411 with respect to the Principle of Security the expert (and Disciple) needs a certain amount of information which is obtained by asking a series of questions (see Figure 5). 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) has enough information about COA411 to make the assessment. As shown in Figure 5, 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.

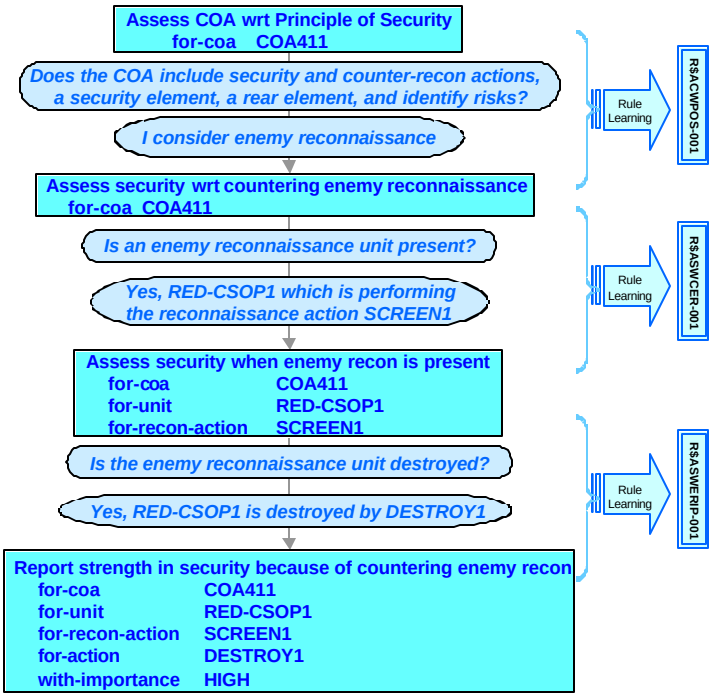


Figure 5: Task reductions and the rules learned from them.

To define a reduction of the current task the expert uses the Example Editor. This, in turn, may invoke the Object Editor, the Feature Editor or the Task Editor, if the specification of the example involves new knowledge elements that are not present in the current ontology. Once the reduction has been defined by the expert the Rule Learner is invoked to learn a general rule from each specific task reduction. Figure 6 shows some details of the process of teaching Disciple.

The left hand side of Figure 6 represents the reasoning process of the expert, the question and the answer being in free natural language format. To learn a rule from this example of task reduction, Disciple needs to find an explanation of why the task from the top of Figure 6 is reduced to the task from the bottom of Figure 6. The explanation to be found, expressed in Disciple's language, is shown in the right hand side of Figure 6. Formally, this explanation consists of a set of paths in Disciple's ontology, each path being a sequence of objects and features.

The information from the explanation is included in the question and the answer from the left hand side of Figure 6. However, the current version of Disciple does not have the ability to understand natural language (although this is a topic of current research). The main role of the question and the answer is to focus the reasoning process of the expert. Also, the domain expert is not a knowledge engineer and therefore cannot be assumed to be able to provide Disciple with the explanation. This would be very difficult for the expert for at least two reasons. First of all there are many hundreds of objects and features names and the domain expert should not be required to learn them. Secondly, the domain expert should not be required to use the formal syntax of Disciple, to be able to correctly define formal explanations.

The current approach to explanation generation relies on the complementary abilities of the domain expert and Disciple. The expert cannot formulate correct explanations, but he can provide some hints to Disciple, for instance by pointing to an object that the explanation should contain. Also, he can recognize a correct explanation piece proposed by Disciple. Disciple, on the other hand, can generate syntactically correct explanation pieces. It can also use analogical reasoning and the hints received from the expert to focus its search and to identify a limited number of plausible explanations from which the expert will have to select the correct ones.

The explanation generation strategy is based on an ordered set of heuristics for analogical reasoning. They 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 for the current example. This cooperative explanation-generation process proved to be very effective, as demonstrated by the successful knowledge acquisition experiment described in section 4.

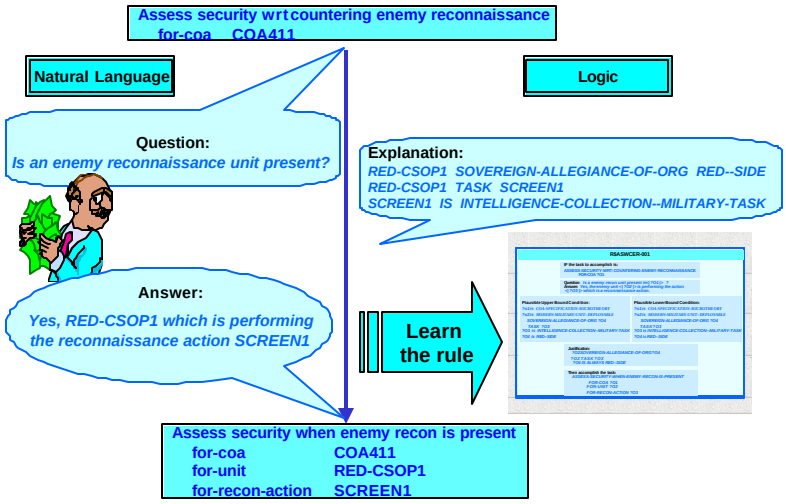


Figure 6: Teaching Disciple to reduce a task.

From the example reduction and its explanation in Figure 6, Disciple automatically generated the plausible version space rule in Figure 7. This is an IF-THEN rule, the components of which are generalizations of the elements of the example in Figure 6. 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. The plausible lower bound condition covers only the example in Figure 6, 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 7, 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 specializations of its conditions.

Let us consider again the task reductions from Figure 5. 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 2. Similarly, the generalizations of the questions and answers from the rules applied to generate a solution are used to produce an abstract justification of the reasoning process.

Comparing the left hand side of Figure 6 (which is defined by the domain expert) with the rule from Figure 7 (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 7. With Disciple, the domain expert (possibly assisted by a knowledge engineer) 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.

R$ASWCER-001

IF the task to accomplish is:
ASSESS-SURPRISE-WRT-COUNTERING-ENEMY-RECONNAISSANCE
FOR-COA ?O1

Question: *Is an enemy recon unit present in ?O1 ?*
Answer: *Yes, the enemy unit ?O2 is performing the action ?O3 which is a reconnaissance action.*

Explanation:
?O2 SOVEREIGN-ALLEGIANCE-OF-ORG ?O4 IS RED--SIDE
?O2 TASK ?O3 IS INTELLIGENCE-COLLECTION--MILITARY-TASK

Plausible Upper Bound Condition:
?O1 IS COA-SPECIFICATION-MICROTHEORY
?O2 IS MODERN-MILITARY-UNIT--DEPLOYABLE
SOVEREIGN-ALLEGIANCE-OF-ORG ?O4
TASK ?O3
?O3 IS INTELLIGENCE-COLLECTION--MILITARY-TASK
?O4 IS RED--SIDE

Plausible Lower Bound Condition:
?O1 IS COA411
?O2 IS RED-CSOP1
SOVEREIGN-ALLEGIANCE-OF-ORG ?O4
TASK ?O3
?O3 IS SCREEN1
?O4 IS RED--SIDE

THEN accomplish the task:
ASSESS-SURPRISE-WHEN-ENEMY-RECON-IS-PRESENT
FOR-COA ?O1
FOR-UNIT ?O2
FOR-RECON-ACTION ?O3

Figure 7: Plausible version space rule learned from the example and explanation in Figure 6.

4 Evaluation of the COA Critiquers and of the Knowledge Acquisition Tools

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 HPKB's annual evaluation. There was a one week dry run evaluation (May 10-15, 1999) of all the COA critiquers that had as a main objective to debug the evaluation mechanics. The actual evaluation took place during the period July 6-16, 1999, and was organized as 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, 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. "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 8 and show that Disciple-COA has obtained the best results out of the four developed critiquers.

Figure 9 compares the recall and the coverage of the developed critiquers for the last three most complex items of the evaluations. For each evaluation 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

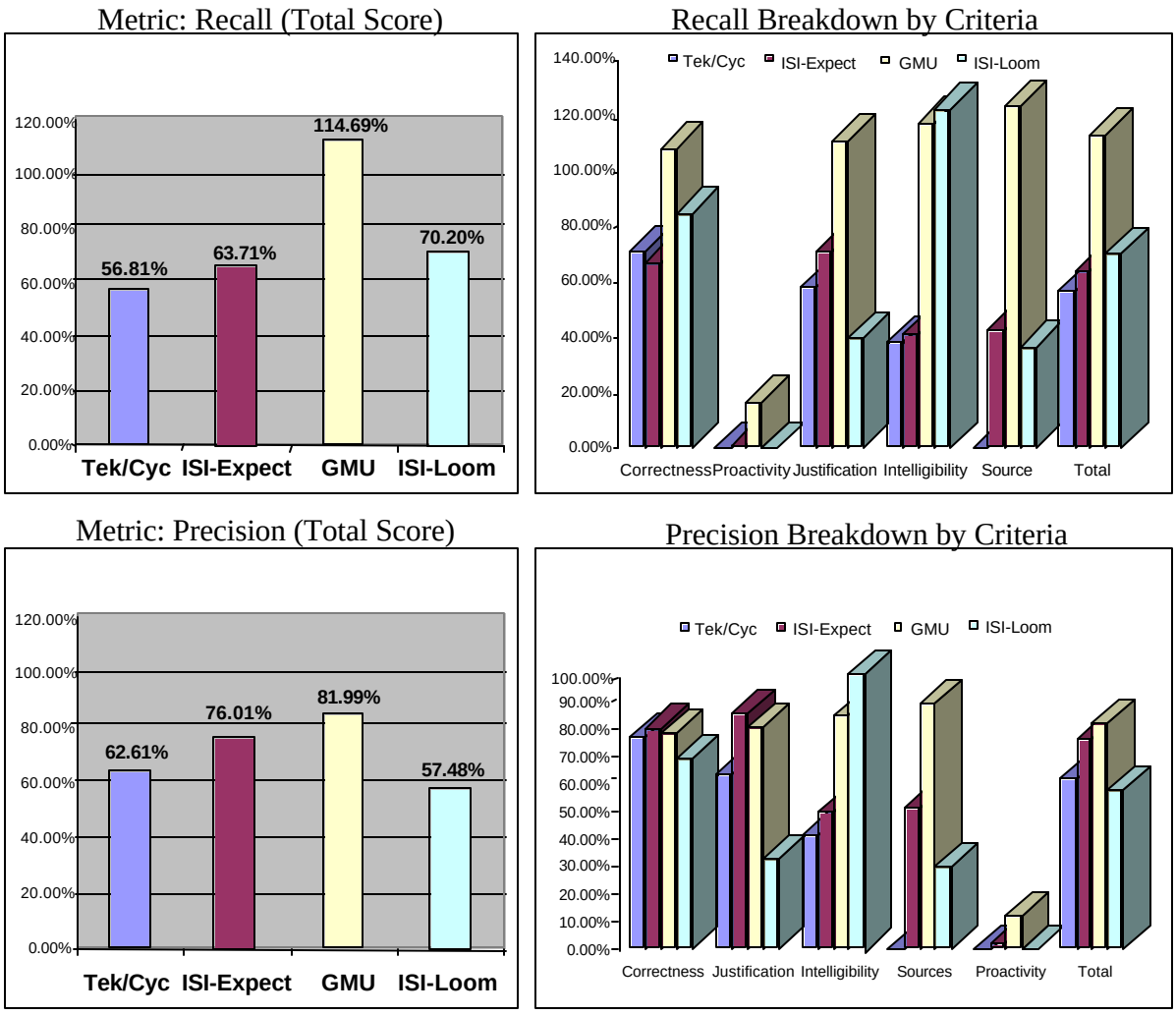


Figure 8: Comparison of the performance of the developed COA critiquers.

for the modification phase. In this case the systems that are above and to the right are superior to the other systems. This graph shows that all the systems increased their coverage during the evaluation, this being one of the aspects tested for each system. In particular, the KB of Disciple increased by 46% during the evaluation period (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.

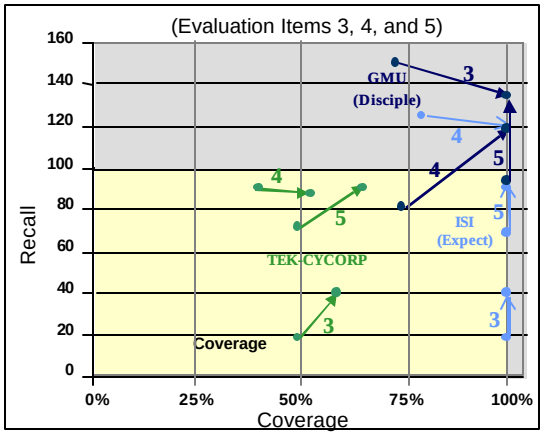


Figure 9: Coverage vs Recall, Pre- and Post-Repair

During August 1999 we conducted a special one week knowledge acquisition experiment with Disciple-COA, at the US Army Battle Command Battle Lab, in Fort Leavenworth, Kansas. In this experiment, four military experts that did not have any prior knowledge engineering experience received around 16 hours of training in Artificial Intelligence and the use of Disciple-COA. They then succeeded in training Disciple to critique COAs with respect to the Principle of Offensive and the Principle of Security, starting with a KB containing the complete ontology of objects and features but not rules. During the training process that lasted around three hours, and without receiving any significant assistance from knowledge engineers, each expert succeeded in extending the KB of Disciple-COA with 28 tasks and 26 rules (approx. 353 simple axioms), following a modeling of the critiquing process (such as the one in Figure 5) that was provided to them at the beginning of the experiment. At the end of the experiment they completed a detailed questionnaire inquiring about their perceptions of the usefulness and usability of the Disciple tool and the Disciple critiquer. An analysis of their answers revealed again very high scores for the Disciple approach (82.39% on the fitness of the Disciple critiquing agent to their organizations, 76.32% in the effect that Disciple-COA would have on their task performance, and 73.72% in system's usability).

5 Conclusions

We have presented an approach to the development of knowledge bases for KB agents and its rapid and successful use for 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 subject matter experts to extend the knowledge base of the critiquer without any or 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 Disciple approach facilitates the process of knowledge base development because it reduces the complex operations that are necessary in order to build a knowledge base to simpler operations. Rather than creating an ontology from scratch, one can import it from a repository of knowledge and update it accordingly. Rather than defining general problem solving rules the expert needs only to provide specific examples because Disciple can generalize them into rules. Rather than creating sentences in an unfamiliar formal language, the domain expert needs only to understand sentences generated by Disciple and select the relevant ones. Finally, rather then providing explanations to the system the expert may only need to provide hints and let the agent find the explanations. As the knowledge

acquisition experiment has demonstrated, even the current version of Disciple allows a subject matter expert to perform several such operations without being assisted by a knowledge engineer. This shows that by further developing this approach it will become possible for domain experts to directly build knowledge bases. 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 use personal computers for text processing or email.

Acknowledgments. This research was supported by AFOSR and DARPA through the grant F49620-97-1-0188, as part of the High Performance Knowledge Bases Program. The evaluation of the COA critiquers 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. Ping Shyr, Bogdan Stanescu, Liviu Panait, Marinel Alangiu and Cristina Cascaval are contributing to the new version of Disciple.

References

Alphatech, Inc. *HPKB Course of Action Challenge Problem Specification*, Burlington, MA, 1999

Boicu M., Wright K., Marcu D., Lee S.W., Bowman M. and Tecuci G., "The Disciple Integrated Shell and Methodology for Rapid Development of Knowledge-Based Agents," in *AAAI-99/IAAI-99 Proceedings*, Intelligent Systems Demonstrations, AAAI Press, Menlo Park, CA. 1999.

Chaudhri, V. K., Farquhar, A., Fikes, R., Park, P. D., and Rice, J. P., OKBC: A Programmatic Foundation for Knowledge Base Interoperability. In Proc. AAAI-98, pp. 600 – 607, Menlo Park, CA: AAAI Press, 1998.

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

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

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

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

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

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

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

Tecuci, G., Boicu, M., Wright, K., Lee, S.W., Marcu, D. and Bowman, M. An Integrated Shell and Methodology for Rapid Development of Knowledge-Based Agents, in in *AAAI-99/IAAI-99 Proceedings*, AAAI Press, Menlo Park, CA. 1999.