The Disciple Integrated Shell and Methodology for Rapid Development of Knowledge-Based Agents

Mihai Boicu, Kathryn Wright, Dorin Marcu, Seok Won Lee, Michael Bowman and Gheorghe Tecuci

Learning Agents Laboratory, Department of Computer Science, MSN 4A5, George Mason University, Fairfax, VA 22030
{mboicu, kwright, dmarcu, swlee, mbowman3, tecuci}@gmu.edu

Abstract

The Disciple Learning Agent Shell (Disciple-LAS) is an integrated set of modules for rapid development of practical end-to-end knowledge-based agents, by domain experts, with limited assistance from knowledge engineers. Disciple-LAS and its associated agent building methodology are presented in (Tecuci et al. 1999). Therefore, in this paper, we introduce two very different agents developed with Disciple-LAS, to show its applicability to a wide range of domains. Then we introduce the different modules that are part of Disciple-LAS, and present their use in the agent building process. Finally we summarize the solutions proposed by the Disciple approach to some of the issues that have been found to be limiting factors in developing knowledge-based agents.

Introduction

Disciple-LAS is an integrated set of modules for rapid development of practical end-to-end knowledge-based agents, by domain experts, with limited assistance from knowledge engineers. It consists of knowledge acquisition, learning and problem solving modules, developed to support the specific Disciple methodology for building an agent (Tecuci et al. 1999). The knowledge base of such an agent has two components: an ontology that defines the concepts from the application domain, and a set of problem solving rules expressed with these concepts. The problem solving approach of an agent built with Disciple-LAS 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 Disciple-LAS 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) the teaching the agent to perform its tasks, teaching that resembles how an expert would teach a human apprentice when solving problems in cooperation.

Disciple-LAS was developed as part of the DARPA's High Performance Knowledge Bases Program (Cohen et al. 1998), and was applied to build two very different agents, a planning agent and a critiquing agent. The planning agent uses hierarchical task decomposition to generate a partially ordered plan of actions for repairing or bypassing some damage to a bridge or road. The building and evaluation of this agent is presented in (Tecuci et al., 1999). The critiquing agent analyses a military course of action (COA) to identify its strengths and weaknesses with respect to the principles of war (e.g. the principles of objective, offensive, mass, economy of force, maneuver, etc.) and the tenets of army operations (e.g. the tenets of initiative, agility, depth, etc.). The critiquing of a COA is also modeled as a task reduction process. For instance, the task of assessing a COA with respect to the principle of objective is reduced to three simpler assessment tasks: 1) assess identification of objective, 2) assess attainability of objective and 3) assess decisiveness of objective. Then each such assessment task is successively reduced to simpler assessment tasks and ultimately reduced to assertions on how the COA conforms to the principle of objective.

Disciple-LAS modules

Figure 1 presents the main modules of Disciple-LAS. They include knowledge import/export modules, ontology management modules, and problem solving and learning modules.

The problem of importing knowledge from an outside knowledge server is reduced to two simpler problems: 1) a translation problem where an external ontology is translated into a Disciple ontology, and 2) an integration problem where the translated ontological knowledge is incorporated into agent's ontology. For the first process one uses either the OKBC protocol to extract knowledge from an OKBC sever, or a translator from KIF (Genesereth and Fikes, 1992) into Disciple. For the second process one uses the ontology management tools of Disciple.

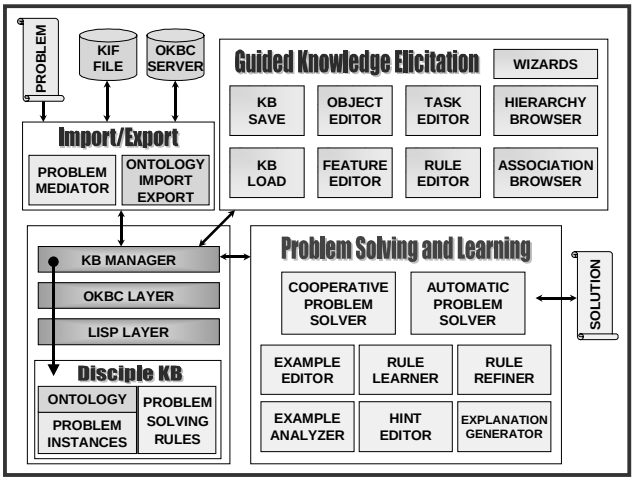

Figure 1: Disciple-LAS modules.

The ontology management tools include a specialized editor for each type of knowledge element to facilitate the interaction with the users. For instance, there is an Object Editor, a Feature Editor, and a Task Editor. We attempt to provide each module with a certain degree of "intelligence", based on "wizards". An important wizard is the Delete Wizard that is automatically invoked whenever the user attempts to delete an element from the KB. This wizard guides the user through a sequence of modifications of the KB that are necessary in order to maintain the consistency of the KB.

The problem solving and learning modules include a Cooperative Step-by-step Problem Solver and an Autonomous Problem Solver, a Rule Learner, and a Rule Refiner, an Example Editor, an Example Analyzer, a Hint Editor, and an Explanation Generator. Their interactions are presented in the following section.

Cooperative problem solving and learning

After an initial ontology has been created, the subject matter expert may start to teach the Disciple agent how solve problems, by following an interaction pattern shown in Figure 2.

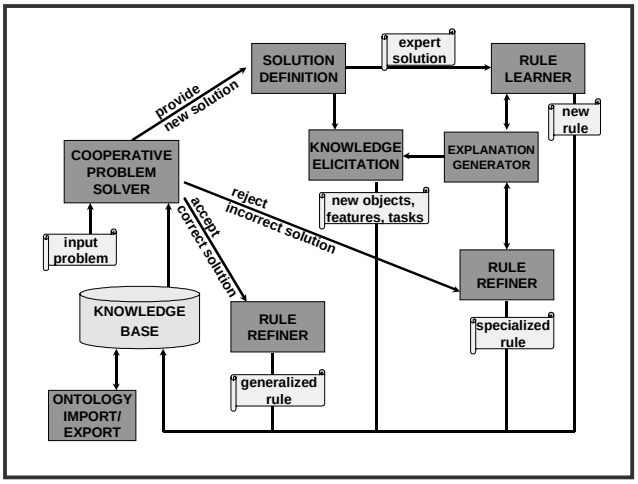

Figure 2: Expert-agent interactions.

The expert invokes the Cooperative Problem Solver, selects or defines an initial task and asks the Disciple agent to reduce 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.

To define a new reduction the expert uses the Example Editor. This, in turn, may invoke 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 ontology. 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 by the expert is correct. The Explanation Generator proposes several plausible explanations from which the expert has to select the correct one. The expert may help the agent to find the correct explanation by providing a hint defined with the Hint Editor. This may again lead to the invocation of the ontology management modules to define any new knowledge base element that is included in the hint.

If the expert accepts a reduction proposed 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 and the Hint Editor being invoked, 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.

Final Remarks

Disciple-LAS provides solutions to some of the issues that have been found to be limiting factors in developing knowledge-based agents. Through ontology import it can reuse previously developed knowledge. The knowledge acquisition and adaptation bottlenecks are alleviated through the use of apprenticeship multistrategy learning methods and a synergistic interaction between the expert and the agent where each does what it can do best, and receives help from the other party. Disciple-LAS contains many general modules that will be part of any Disciple agent, but it also allows for the customization of some of its modules, trying to achieve a suitable balance between reusing general modules and developing domain specific ones. Finally, Disciple-LAS and the Disciple agents are portable, being implemented in JAVA and Common LISP.

Acknowledgments. This research was supported by the AFOSR grant F49620-97-1-0188, as part of the DARPA's High Performance Knowledge Bases Program.

References

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

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

Genesereth M.R. and Fikes R.E. 1992. Knowledge Interchange Format, Version 3.0 Reference Manual. KSL-92-86, Knowledge Systems Laboratory, Stanford University.

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., *An Integrated Shell and Methodology for Rapid Development of Knowledge-Based Agents*, In *Proc. AAAI-99*, Menlo Park, CA: AAAI Press.