A Mixed-Initiative Approach to Rule Refinement for Knowledge-Based Agents

Cristina Boicu, Gheorghe Tecuci, Mihai Boicu

Learning Agents Center, Department of Computer Science, MSN 4A5,
George Mason University, 4400 University Dr., Fairfax, VA 22030, Phone (703) 993-4669
{ccascava, tecuci, mboicu}@gmu.edu; http://lac.gmu.edu

Abstract

This paper presents a mixed-initiative approach to rule refinement in which a subject matter expert collaborates with a learning agent to refine the agent's knowledge base. This approach is implemented in the Disciple learning agent shell and has been evaluated in several agent training experiments performed by subject matter experts at the US Army War College.

1 Introduction

Our research addresses the problem of developing knowledge-based agents that incorporate the subject matter expertise of human experts. Our approach is to develop a learning and problem solving agent, called Disciple, which can be directly taught by a subject matter expert by explaining it how to solve specific problems, and by critiquing its attempts to solve new problems (Tecuci 1998).

The knowledge base of a Disciple agent is structured into an object ontology that contains a hierarchical description of the objects and features from an application domain, and a set of task reduction rules and solution composition rules expressed with these objects.

To teach the Disciple agent, the expert defines specific problems and helps the agent to understand each reasoning step toward their solutions. From each new reasoning step and its explanation, Disciple learns a general problem solving rule. However, the rules learned by the agent are generally incomplete and need to be refined. The main reason for this situation is that subject matter experts usually express their knowledge informally, in natural language, using visual representations and common sense reasoning, often omitting essential details that are implicit in human communication. By contrast, the knowledge of an agent must be represented in a formal, precise and fairly complete way. The consequence of this mismatch is that an expert's knowledge is only partially expressed in the agent's knowledge base. Therefore the agent's problem solving rules need to be continuously refined in order to better characterize the subtle distinctions that experts make in their domain.

Rule refinement involves a mixed-initiative interaction, because neither the expert nor the agent can solve this problem independently. On one hand, the expert has the knowledge that needs to be incorporated into the agent's knowledge base, but he is not familiar with the agent's formal knowledge representation. On the other hand, the agent does not know the subtle distinctions from the application domain, but it incorporates methods to acquire and formalize the expert's knowledge.

In this paper we present a mixed-initiative assistant that supports a subject matter expert in refining the rules from the agent's large knowledge base. In this process the responsibility is divided between the expert and the agent, such that each performs the tasks for which he has the best capability. The next section presents a brief overview of the rule learning and refinement process. Then the following sections provide more details on some of the developed methods.

2 Rule Learning and Refinement

The rule learning and refinement process in Disciple is based on the observation that it is difficult for a subject matter expert to work directly with formal rules. It is much easier for the expert to analyze specific examples, to accept them as correct, or to reject them as incorrect, and to provide explanations of his decisions.

In order to illustrate the rule learning process let us consider the development of an agent that can assist a student to select a PhD advisor. A fragment of the object ontology of this agent is shown in Figure 1. This object ontology will be used as a generalization hierarchy for learning.

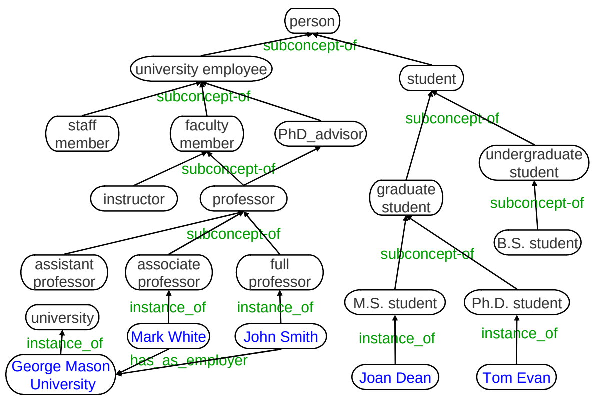

Figure 1: Ontology fragment

To teach this agent, the expert formulates a problem solving task, such as "Determine a PhD advisor for Tom Evan." Then, using the task reduction paradigm, the expert successively reduces this task to simpler tasks, guided by questions and answers, as illustrated by the task reduction step from Figure 2.

Task: Determine whether John Smith can be a PhD advisor for Tom Evan in Artificial Intelligence.

Question: *Is John Smith likely to stay on the faculty of George Mason University for the duration of Tom Evan's dissertation?*
Answer: *Yes, because John Smith has a tenured position.*

Subtask: Determine whether John Smith would be a good PhD advisor for Tom Evan in Artificial Intelligence.

Figure 2: Task reduction example

From each such task reduction step, Disciple learns a general task reduction rule, by using a mixed initiative approach, as described in (Tecuci et al. 2004). Figure 3, for instance, shows the rule learned from the example in Figure 2. These rules learned by Disciple have an IF-THEN part that preserves the expert's language from the example, and a main condition which specifies the values of the rule's variables for which the rule generates a correct task reduction step.

IF: Determine whether ?O1 can be a PhD advisor for ?O2 in ?O3

Question: *Is ?O1 likely to stay on the faculty of ?O4 for the duration of ?O2 's dissertation?*
Answer: *Yes, because ?O1 has a ?O5*

THEN: Determine whether ?O1 would be a good PhD advisor for ?O2 in ?O3

MAIN CONDITION

?O1 is	*PUB (PhD_advisor)*	*PLB (PhD_advisor)*
	has_as_employer ?O4	
	has_as_position ?O5	
?O2 is	*PUB (person)*	*PLB (PhD_student)*
?O3 is	*PUB (research_area)*	*PLB (Artificial_Intelligence)*
?O4 is	*PUB (employer)*	*PLB (university)*
?O5 is	*PUB (position)*	*PLB (tenured_position)*

Positive Example: (?O1=John_Smith ?O2=Tom_Evan ?O3=Artificial_Intelligence ?O4=George_Mason_University ?O5=tenured_position)

Figure 3: Rule learned from the example in Fig. 2

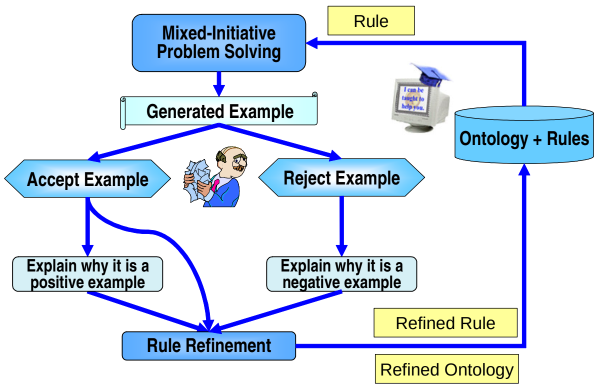

Figure 4: The rule refinement process

The rule in Figure 3, however, is only partially learned. Therefore, instead of a single applicability condition it includes a version space for the final condition to be learned, based on additional examples and their explanations (Tecuci et al. 2002). This version space is characterized by a plausible upper bound (PUB), which is based on a maximal generalization of the example in Figure 2, and by a plausible lower bound (PLB), which is based on a minimal generalization of the example that does not contain any instance (Tecuci et al. 2005). For example, the PLB condition requires ?O1 to be a PhD advisor who has as employer ?O4 (that has to be a university) and has as position ?O5 (that has to be a tenured position). In addition, ?O2 has to be a PhD student and ?O3 has to be Artificial Intelligence.

Although partially learned, the rule in Figure 3 may be used in problem solving when either its PLB condition or its PUB condition is satisfied, as summarized in Figure 4. Indeed, let us assume that, after having learned the rule from Figure 3, Disciple attempts to "Determine whether Mark White can be a PhD advisor for Tom Evan in Information Security." Because the PUB condition of the rule from Figure 3 is satisfied for this task, Disciple applies it and proposes the reduction to the expert: "Determine whether Mark White would be a good PhD advisor for Tom Evan in Information Security." If the expert accepts this solution, then the PLB condition is generalized to cover it. However, the expert rejects it, explaining that "Mark White is likely to move to Stanford University." Consequently, Disciple adds an except-when condition to the rule which takes the form shown in Figure 5. From now on the rule will be applicable if its main condition is satisfied and its except-when condition is not satisfied.

During rule refinement the ontology may also need to be extended with new elements present in the explanations of the examples. In this mixed-initiative process, the expert and the agent share representations, communicate naturally through the use of specific examples and explanations, coordinate their actions, take initiative and release control to identify the necessary refinements for the rules, as discussed in the next section.

IF: Determine whether ?O1 can be a PhD advisor for ?O2 in ?O3

Question: *Is ?O1 likely to stay on the faculty of ?O4 for the duration of ?O2 's dissertation?*
Answer: *Yes, because ?O1 has a ?O5*

THEN: Determine whether ?O1 would be a good PhD advisor for ?O2 in ?O3

MAIN CONDITION

?O1	*is*	*PUB (PhD_advisor)*	*PLB (PhD_advisor)*
	has_as_employer ?O4		
	has_as_position ?O5		
?O2	*is*	*PUB (person)*	*PLB (PhD_student)*
?O3	*is*	*PUB (research_area)*	*PLB (Artificial_Intelligence)*
?O4	*is*	*PUB (employer)*	*PLB (university)*
?O5	*is*	*PUB (position)*	*PLB (tenured_position)*

EXCEPT WHEN CONDITION

?O1	*is*	*PUB (person)*	*PLB (PhD_advisor)*
	is_likely_to_move_to ?O6		
?O6	*is*	*PUB (employer)*	*PLB (university)*

Positive Example: (?O1=John_Smith ?O2=Tom_Evan ?O3=Artificial_Intelligence ?O4=George_Mason_University ?O5=tenured_position)
Negative Example: (?O1=Mark_White ?O2=Tom_Evan ?O3=Information_Security ?O4=George_Mason_University ?O5=tenured_position ?O6=Stanford_University)

Figure 5: Rule refined with a negative example

3 A Mixed-Initiative Approach to Rule Refinement

The refinement of the agent's rules by a subject matter expert is a difficult and complex process, especially because the expert has no or little knowledge engineering experience. Moreover, an expert does not have a complete understanding of how the knowledge is represented in the agent's knowledge base.

Therefore, it is difficult for a subject matter expert to work directly with formal rules, and to analyze their correctness. However, it is much easier for the expert to critique specific examples and to provide a justification of his critique. Figure 6 summarizes a mixed-initiative approach in which the Disciple agent collaborates with the subject matter expert in refining the rules from its knowledge base. This process involves several component agents that closely interact with each other to help the expert in the refinement of the rules.

3.1 Discovering Incomplete Rules

In expressing their knowledge, domain experts use common sense and often omit details that are implicit in human communication. Therefore an agent will learn rules from partially explained examples. To alleviate this problem we have developed the Rule Analyzer which continuously analyzes the rules and guides the expert to provide more complete explanations (see phase 1 in Figure 6: Discovering Incomplete Rules).

The Rule Analyzer agent takes the initiative to analyze the rules after each modification, to determine whether they need further refinement. First, the Rule Analyzer performs a structural analysis of the rule, checking if the values of its output variables (i.e. the variables from the question, answer and THEN tasks) can be obtained from the values of the variables of the IF task. Then, the Rule Analyzer checks how the rule will apply during problem solving and determines if there are too many instances of the rule and which are the under-constrained variables.

The Rule Analyzer combines the results of the performed checks and displays a list of plausible problems and their suggested solutions to the expert. This process keeps the expert informed and assures that a better rule is learned from the very beginning. The Rule Analyzer is invoked both during the initial phase of rule learning and during rule refinement based on additional positive and negative examples.

3.2 Guiding the Rule Refinement Process

When the Disciple agent solves a given problem it shows to the expert the entire reasoning tree. However, the reasoning tree is generally very large, and can contain hundreds, even thousands of problem solving steps (a problem solving step corresponds to the application of a rule in a specific situation).

To address this problem, we have developed several rule refinement wizards that guide the expert in understanding, navigating and refining a complex reasoning tree, by focusing the expert on those steps of the agent's reasoning process that require expert's analysis. These are steps generated by using highly incomplete rules, or the plausible upper bound conditions of partially incomplete rules (see phase 2 in Figure 6: Guiding the Rule Refinement Process).

Analyzing and Refining a Reasoning Sub-Tree

During the refinement of a complex reasoning tree, the expert usually has to navigate through the entire tree to see if other reasoning steps need refinement. This process is very tedious and time consuming, and the expert needs to pay a lot of attention to follow each node in the tree, in order to identify the next steps that need to be refined.

We have developed a wizard that helps the expert to analyze a reasoning sub-tree of a problem solving node, after it was refined. After the expert critiques a problem solving step (example), the Analyze Sub-Tree wizard will indicate to the expert the next step that needs to be refined in that sub-tree. The wizard checks each of these reasoning steps and signals to the expert the problems found, helping him in the refinement process.

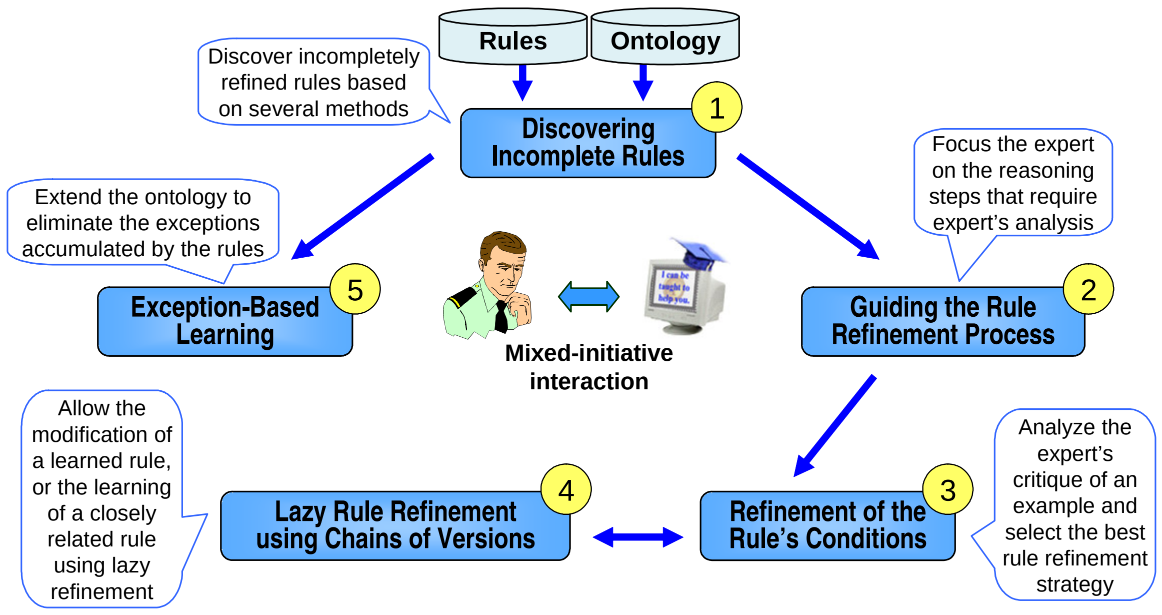
Rules
Ontology
Discover incompletely refined rules based on several methods
Discovering Incomplete Rules
1
Focus the expert on the reasoning steps that require expert's analysis
Extend the ontology to eliminate the exceptions accumulated by the rules
Exception-Based Learning
5
Mixed-initiative interaction
Guiding the Rule Refinement Process
2
Allow the modification of a learned rule, or the learning of a closely related rule using lazy refinement
Analyze the expert's critique of an example and select the best rule refinement strategy
Lazy Rule Refinement using Chains of Versions
4
Refinement of the Rule's Conditions
3

Figure 6: A Mixed-Initiative Integrated Approach to Rule Refinement

This significantly reduces the time spent by the expert searching for the steps that need to be refined, and also assures that reasoning steps that need refinement will not be omitted, offering to the expert an interactive framework that guides him through this process of rule refinement.

Another feature of the Analyze Sub-Tree wizard consists in keeping the current state of refinement in which the expert is working. For instance, the expert may inspect another reasoning step, to review some information, and then may return to the interrupted stage, to continue the refinement from where it was left. We plan to extend this wizard by ordering the suggestions made to the expert based on their expected impact during rule refinement, showing first the most critical reasoning steps that must be critiqued by the expert.

Analyzing Other Applications of a Rule

After the expert analyzes an example generated by a rule, and the system refines the rule based on the example and its explanation of success or failure, the reasoning tree is dynamically updated and the rule is applied to other reasoning steps.

We have developed a wizard to guide the expert in analyzing the similar cases from the current reasoning tree, facilitating the refinement of the applied rule. The Rule Applications wizard conducts an analysis of the rule corresponding to the example just refined and suggests the next application of the rule that needs to be analyzed by the expert, indicating the identified problems and proposing suggestions for refinement.

We plan to extend this wizard by taking into account the level of similarity between the current example and the proposed similar cases. This option will be very useful when there are many similar cases, and we would prefer to show just a few significantly different examples which are representative for the entire set.

Rule Refinement Using Sub-Tree Validation

We are also developing a wizard to allow the expert to analyze a reasoning sub-tree of a node and specify that all its reasoning steps are correct. As a consequence, all the applicable rules will be generalized with the corresponding examples. The current Sub-Tree Validation has the disadvantage that it will not guarantee that the expert had the opportunity to see the content of the validated sub-tree.

In order to constrain and at the same time to help the expert in analyzing the desired sub-tree, this sub-tree will be extracted and shown in a separate viewer, then the expert will be asked to confirm it. We propose to develop a method that will analyze the entire sub-tree and determine if it is suitable for automatic validation, signaling warnings about potential problems identified by the Rule Analyzer.

3.3 Refinement of the Rule's Conditions

As shown in Figure 5, a rule has a complex structure. The rule is applied when its main condition is satisfied and none of its except-when conditions are satisfied. When the expert selects a task reduction step (example) to analyze, the Explanation Generation agent interacts with him to critique the example and then the Rule Refinement agent updates the rule accordingly.

For instance, when the expert rejects a reduction generated by a rule and provides explanations of why the reduction is wrong, the agent must refine the conditions of the rule (see phase 3 in Figure 6: Refinement of the Rule's Conditions). One strategy is to specialize the upper bound of the main condition to no longer cover that reduction. Another strategy is to generalize the lower bound of one of the except-when conditions to cover that reduction. Yet another strategy is to create a new except-when condition. If the types of explanations elicited from the expert do not allow the agent to choose between competing strategies, the Rule Refinement agent uses a lazy refinement method, postponing the decision until more cases are analyzed by the expert, as discussed in the next section.

3.4 Lazy Rule Refinement

After analyzing a task reduction step generated by the agent, the expert may decide to update it. If the expert adds an explanation to the task reduction step then the agent refines the corresponding rule with a generalization of that explanation. However, if the expert deletes an explanation, or changes one of the rule's tasks, the question, or the answer, then it is not clear whether the rule should be updated, or a new rule should be learned from the modified example. All depends on whether the modification makes sense for the previous examples from which the rule was learned (Boicu et al. 2005).

However, many of these examples may not be accessible in the context of the current problem and situation. Therefore, we have developed a lazy rule refinement method in which the agent creates a new version of the rule corresponding to the modified example, but it also keeps the old, unchanged version, linked to this new version. In order to avoid the generation of very similar solutions by different versions of a rule, the agent generates first the solutions corresponding to the newest version. Solutions corresponding to the older versions are considered only if they are different from those generated by the more recent versions of the rule.

If in a future refinement session the expert confirms an example generated by a previous version of the rule, then this becomes an independent rule and is removed from the linked list. On the other hand, if the examples generated by the previous versions are rejected, they will be used to specialize the conditions of these rule versions. When the conditions of these previous rule versions become empty, the rules are deleted from the knowledge base.

This lazy refinement method allows the modification of a learned rule, or the learning of a closely related rule, without requiring the expert to perform an analysis of the rule's representative examples at the time of the modification. Instead, this analysis is postponed until the agent applies the rule in problem solving (see phase 4 in Figure 6: Lazy Rule Refinement Using Chains of Versions).

3.5 Exception-Based Learning

For a complex application domain, some of expert's knowledge may not be expressible in the agent's representation language. As a consequence, the rules learned by the agent may accumulate exceptions (Tecuci 1998; Wrobel 1994). These exceptions may indicate missing or partially represented ontological knowledge (see Figure 1), and may be used to extend the agent's representation language, in order to better characterize the subtle distinctions that experts make in their domain.

To address this issue, we have developed an Exception-Based Learning Assistant that interacts with the subject matter expert to comparatively analyze the negative exceptions and the positive examples of a rule, in order to discover extensions to the ontology (such as new features and/or new facts of the form "object feature value") that will eliminate the exceptions of the rule (Boicu and Tecuci, 2004).

These representation extensions will lead to the improvement of the rules and the elimination of their exceptions (see phase 5 in Figure 6: Exception-Based Learning).

The component agents described above are integrated into the Rule Refinement module. They complement each other and create an integrated mixed-initiative approach to the rule refinement problem in an evolving representation space, resulting in improved problem solving rules, which will assure a higher degree of correctness of the solutions generated by the agent.

4 Conclusions and Future Research

The knowledge-base refinement problem is addressed by several systems to modify an initial imperfect knowledge-base to become consistent with empirical data, like KR-FOCL (Pazzani and Brunk 1991), EITHER (Mooney and Ourston 1994), ODYSSEUS (Wilkins 1990), KRUST (Craw 1997), MOBAL (Wrobel 1994), TEIRESIAS (Davis 1982). Most of these systems correct propositional Horn-clause theories, by adding or deleting rule antecedents, and by learning or deleting rules, to correctly classify a given set of examples and they are mostly automatic.

However, in this paper we have presented a mixed-initiative approach to this problem, in which the expert and the agent collaborate to make the necessary refinements to the rules learned by the agent assuring a better accuracy and problem solving performance.

We plan to improve the mixed-initiative interaction between the component agents involved during the rule refinement process and the expert, such that the process is more natural and the expert is offered more guidance during the most difficult phases.

We also plan to learn a user profile in which to incorporate the user's knowledge engineering experience, so that the agents can adapt their behavior using this information. The profile will be incrementally developed to include also the user's problem solving knowledge, his assumptions, preferences and biases which can be acquired from the user or inferred from his behavior.

The user's knowledge engineering experience will have an important role for the type of operations that the user may perform in the rule refinement process. There will be a limited type and number of operations that a subject matter expert with no knowledge engineering experience can perform, while a user with more knowledge engineering experience will be able to perform more difficult refinement operations.

For a subject matter expert with no knowledge engineering experience, we plan to use a strategy that hides all the formal aspects, and concentrates only on judging specific examples of a rule, based on which the agent refines the rule internally. However, the disadvantage of such a strategy is that it generally requires more examples, and a longer refinement time. Therefore, for a user who has more knowledge engineering experience the refinement strategies can be more complex and should allow a direct interaction with the rule that generated the analyzed example. Such strategies reduce the refinement time, but may generate errors if incorrectly used.

We also plan to develop methods for learning meta-rules that will represent the user's preferences among competing applicable rules. This feature will be very useful when the agent's knowledge base is very large and many rules are applicable for a given input task.

Preliminary versions of the methods discussed in this paper have been implemented in the Disciple system, and have been successfully evaluated in several knowledge acquisition experiments performed at the US Army War College in the context of the center of gravity determination and intelligence analysis (Tecuci et al. 2005).

Acknowledgements. This research was sponsored by several US Government agencies, including DARPA, AFRL, AFOSR, and US Army War College.

References

Boicu, C.; Tecuci, G. and Boicu, M. 2005. Rule Refinement by Domain Experts in Complex Knowledge Bases. In *Proceedings of the Twentieth National Conference of Artificial Intelligence* (AAAI-2005). July 9-13, 2005, Pittsburgh, Pennsylvania.

Boicu, C. and Tecuci, G. 2004. Mixed-Initiative Ontology Learning. In *Proceedings of the 2004 International Conference on Artificial Intelligence* (ICAI-2004). IEEE Computer Society, Los Alamitos, California.

Craw, S; Boswell, B and Rowe, R. 1997. Knowledge Refinement to Debug and Maintain a Tablet Formulation System. In *Proceedings of the 9TH IEEE International Conference on Tools with Artificial Intelligence*, pp. 446–453, IEEE Press.

Davis, R. 1982. Teiresias. In A. Barr and E. A. Feigenbaum, editors, *The Handbook of Artificial Intelligence Volume 2*, pages 87–101. Morgan Kauffman: Los Altos, CA.

Mooney, R and Ourston, D. 1994. A Multistrategy Approach to Theory Refinement. In *Machine Learning: A Multistrategy Approach*, Vol IV, R.S. Michalski and G. Tecuci (Eds.), pp.141-164. Morgan Kaufman, San Mateo, CA.

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

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

Tecuci, G.; Boicu, M.; Marcu, D.; Stanescu, B.; Boicu, C. and Barbulescu, M. 2004. Parallel Knowledge Base Development by Subject Matter Experts. In *Proceedings of the 14th Int. Conference on Knowledge Engineering and Knowledge Management* (EKAW 2004). Springer-Verlag.

Tecuci, G.; Boicu, M.; Boicu, C.; Marcu, D.; Stanescu, B. and Barbulescu, M. 2005. The Disciple-RKF Learning and Reasoning Agent. To appear in the *Special Issue on Learning to Improve Reasoning of the Computational Intelligence Journal.*

Wrobel, S. 1994. *Concept Formation and Knowledge Revision.* Dordrecht, Netherlands: Kluwer Academic Publishers.

Wilkins, D. 1990. Knowledge base refinement as improving an incorrect and incomplete domain theory. In Y. Kodratoff and R. S. Michalski, editors, *Machine Learning*, Vol. III, pp. 493–513. Morgan Kaufmann, San Mateo, CA.