In Proceedings of the Sixth International Conference on Machine Learning and Applications (ICMLA'07)
Cincinnati, Ohio, December 13-15, 2007.

Learning Complex Problem Solving Expertise from Failures

Cristina Boicu, Gheorghe Tecuci, Mihai Boicu
Learning Agents Center, Department of Computer Science, MS 6B3
George Mason University, 4400 University Drive, Fairfax, VA 22030-4444
{ccascava, tecuci, mboicu}@gmu.edu, http://lac.gmu.edu

Abstract

Our research addresses the issue of developing knowledge-based agents that capture and use the problem solving knowledge of subject matter experts from diverse application domains. This paper emphasizes the use of negative examples in agent learning by presenting several strategies for capturing expert's knowledge when the agent fails to correctly solve a problem. These strategies have been implemented into the Disciple learning agent shell and used in complex application domains such as intelligence analysis, center of gravity determination, and emergency response planning.

1. Introduction

We address the development of knowledge-based agents that can rapidly capture and use the problem solving knowledge of subject matter 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 giving the agent specific problem solving examples and explanations, and correcting its behavior when it attempts to solve new problems [1]. We have developed applications in several domains, including intelligence analysis, center of gravity determination, emergency response planning, course of action critiquing, and PhD advisor selection.

In this paper we will consider the intelligence analysis domain. For this domain, we have developed an agent to assist an intelligence analyst in assessing the likelihood of various hypotheses, such as whether a specific terrorist group has nuclear weapons. To solve this problem, we have developed a systematic approach to hypothesis and source analysis that is both natural for an analyst and appropriate for an automated agent. This approach is based on Disciple's task reduction reasoning paradigm [2]. To illustrate it, let us consider that the analyst specifies an initial intelligence analysis task to solve (e.g. "Assess whether Terrorist_Group_T has nuclear weapons"). This complex task is successively reduced to simpler tasks, guided by questions and answers, until elementary solutions are reached. Then these solutions are successively composed, until the solution to the initial task is found, as shown in Figure 1. Generally such a reasoning tree contains thousands of reasoning steps. At the top of the tree, the initial hypothesis is reduced to simpler hypotheses, until some elementary hypotheses are reached. Then potentially relevant pieces of evidence are identified and analyzed to determine to what extent they favor or disfavor the corresponding hypotheses. The analysis takes into account the chains of custody of the pieces of evidence, as well as the competence and the credibility of the corresponding primary and intermediary sources.

Each task reduction step from the reasoning tree is obtained by applying reasoning rules from the agent's knowledge base. These rules have applicability conditions expressed with the terms from the agent's object ontology, which is a hierarchical representation of the objects and features from the application domain.

An example of a task reduction rule is shown in Figure 2. This rule has an IF-THEN structure containing an informal section (shown in the top part of Figure 2) which preserves the expert's natural language and is used in the agent-user communication. The rule also contains a formal section (shown in the bottom part of Figure 2) which is used in the internal reasoning of the agent. This particular rule has a main applicability condition (which has to be satisfied by some instances from the object ontology, in order for the rule to be applicable) and two except-when conditions (which should not be satisfied by any instance in the object ontology).

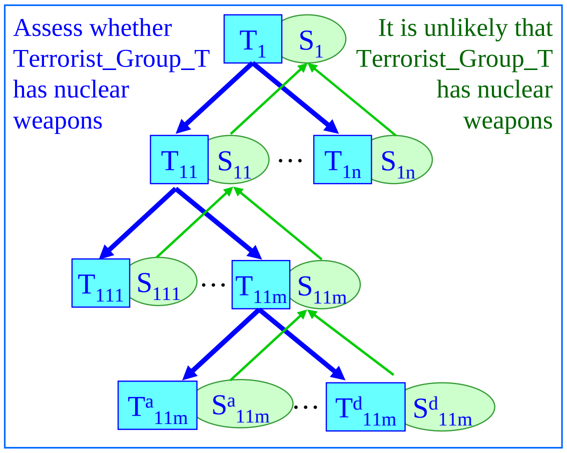

Figure 1: Task reduction paradigm

In Proceedings of the Sixth International Conference on Machine Learning and Applications (ICMLA'07)
Cincinnati, Ohio, December 13-15, 2007.

Moreover, this rule is only partially learned and each condition is represented by two plausible bounds, a plausible upper bound (PUB) condition, and a plausible lower bound (PLB) condition. During further learning, the rule may be refined with additional except-when conditions, and the two bounds of each condition will converge toward one another [2].

One characteristic aspect of our approach is that these rules are learned by the agent through a mixed-initiative interaction, based on specific examples of task reduction steps and their explanations provided by the expert. For instance, when the agent does not know how to solve a task, the expert may provide one or more reduction steps. From these steps and their explanations, the agent will learn general rules. These rules will be used in future problem solving situations, when either their PLB conditions or their PUB conditions are satisfied. This will allow the agent to collaborate with the expert in solving new problems, and to refine the rules based on the correctness of the reductions generated by them (where the correct reductions are used as positive examples of the rules, and incorrect ones as negative examples) [3]. In this paper we will concentrate on the problem of using the negative examples in the rule refinement process.

IF
Assess whether there are states willing to sell nuclear weapons to ?O1

Question: *Which is a nuclear state that is not an enemy of ?O1 and does not oppose the proliferation of nuclear weapons?*
Answer: *?O2*

THEN
Assess whether ?O2 is willing to sell nuclear weapons to ?O1

MAIN CONDITION

PUB Condition		**PLB Condition**	
?O1	is actor	?O1	is terrorist_group
?O2	is nuclear_state	?O2	is nuclear_state

EXCEPT-WHEN CONDITION

PUB Condition		**PLB Condition**	
?O1	is actor	?O1	is terrorist_group
?O2	is actor	?O2	is nuclear_state
	perceives_as_enemy ?O1		perceives_as_enemy ?O1

EXCEPT-WHEN CONDITION

PUB Condition		**PLB Condition**	
?O1	is actor	?O1	is terrorist_group
?O2	is actor	?O2	is nuclear_state
	degree_of_opposition_to_nuclear weapons_proliferation ?S1		degree_of_opposition_to_nuclear weapons_proliferation ?S1
?S1	is [very-low, very high]	?S1	is [very-high, very-high]

Positive Example: (?O1=Terrorist_Group_T ?O2=State_A)

Negative Example: (?O1=Terrorist_Group_T ?O2=State_B)

Negative Example: (?O1=Terrorist_Group_T ?O2=State_C)

Figure 2: A task reduction rule

Section 2 will present the mixed-initiative problem solving process. Section 3 will describe various rule refinement strategies based on negative examples, using an illustration from the intelligence analysis domain. Section 4 will present experimental results and Section 5 will conclude with a discussion of related research.

2. Mixed-Initiative Problem Solving

During the mixed-initiative problem solving process, the expert and the agent solve together a problem. Initially the agent has the initiative, searches for applicable rules in its knowledge base, and generates plausible solutions of the problem. Next, the expert may take the initiative to critique the solutions proposed by the agent. We will distinguish between the following situations:

1) The expert accepts a reasoning step performed by the agent, using a given problem solving rule. a) If the expert does not provide any additional explanation, then the agent will generalize the applicability condition of the corresponding rule in order to cover this example and it will keep the confirmed reasoning step as a positive example for this rule. b) If the expert provides more explanations for this reasoning step, then these explanations will be used by the agent to refine the main applicability condition of this rule. The expert may even accept an entire reasoning tree, in which case all the rules used in that tree will be refined with their corresponding instantiations used as positive examples.

2) The expert rejects a reasoning step performed by the agent. a) If the expert provides an explanation why it is incorrect, then the agent will keep this reasoning step as a negative example and the explanation will be used by the agent to specialize the rule to no longer cover this negative example. b) If the expert does not provide any explanation, then the agent will keep this reasoning step as a negative exception, which will need to be analyzed later by the expert during the exception-based learning and refinement process [4].

3) The expert may also extend the reasoning tree with new steps that solve the current problem. Then, the expert will help the agent to understand these reasoning steps, based on which the agent will learn general problem solving rules.

During this process, the ontology may also be extended with new elements in order to allow the refinement of the rules to correctly cover the generated positive examples and to not cover the negative examples.

In Proceedings of the Sixth International Conference on Machine Learning and Applications (ICMLA'07)
Cincinnati, Ohio, December 13-15, 2007.

3. Rule Refinement Strategies Based on Negative Examples

When the agent generates a reasoning step which is rejected by the expert, the agent must refine the condition of the rule, to no longer cover this negative example. However, there are several strategies to modify the rule's condition to no longer cover this incorrect reduction. 1) One strategy is to specialize the main condition to no longer cover the example. 2) Another strategy is to generalize one of the rule's except-when conditions to cover the example. 3) Yet another strategy is to add a new except-when condition to the rule.

There are many difficulties in the refinement of a rule based on a negative example. One difficulty is the risk of over specialization of the rule. To avoid this risk we use except-when plausible version space conditions to better approximate the knowledge to be learned from the negative examples. These conditions have a conservatively generated plausible lower bound, covering only the examples that are very similar with the negative examples rejected by the expert. When a new example is rejected, the lower bound is minimally generalized to cover also this new example. This condition has also an upper bound, which is the maximal generalization of the rejected examples. However, this bound will probably cover also correct examples. In order to allow the refinement of existing except-when conditions of a plausible version space rule, the problem solver needed to be modified to allow the generation of some examples that are covered by the upper bound of except-when conditions. Disciple uses these examples to further refine the rules.

Another difficulty of rule refinement based on negative examples is the risk of proliferation of unrefined except-when conditions. This means that for each new incorrect example the user will learn a new except-when condition instead of selecting other refinement strategy (i.e. specializing the main condition or an except-when condition). In order to avoid this, the agent will prefer the strategies to refine existing conditions and if those are not acceptable, it will learn a new except-when condition.

To illustrate the rule refinement process, let us consider that the expert provides the reduction step shown in Figure 3, for assessing whether there are states willing

We need to:

Assess whether there are states willing to sell nuclear weapons to Terrorist_Group_T

Question: *Which is a nuclear state ?*

Answer: *State_A*

THEN we need to:

Assess whether State_A is willing to sell nuclear weapons to Terrorist_Group_T

Figure 3: An example of a task reduction step

IF
Assess whether there are states willing to sell nuclear weapons to ?O1

Question: *Which is a nuclear state ?*
Answer: *?O2*

THEN
Assess whether ?O2 is willing to sell nuclear weapons to ?O1

MAIN CONDITION

PUB Condition	**PLB Condition**
?O1 is actor	?O1 is terrorist_group
?O2 is nuclear_state	?O2 is nuclear_state

Positive Example: (?O1=Terrorist_Group_T ?O2=State_A)

Figure 4: The rule learned from the example in Figure 3

to sell nuclear weapons to a specific terrorist group T. Based on this initial example and its explanation ("State_A is nuclear_state"), the agent learns the general task reduction rule shown in Figure 4.

The agent applies this learned rule in problem solving and generates other task reduction steps to be analyzed by the expert. The expert notices a solution where a state (State_B) which is an enemy of Terrorist_Group_T, would sell weapons to it, and rejects it as incorrect. This initiates an explanation generation process in which the expert collaborates with the agent to justify why this example is incorrect: "State_B perceives as enemy Terrorist_Group_T", therefore it would not be willing to sell nuclear weapons to this group. Based on this explanation, the agent generates a plausible version space except-when condition and adds it to the rule. The addition of the except-when condition specializes both the lower and upper bounds of the rule's overall applicability condition.

The agent re-applies the refined rule. This time the expert notices a solution where a state (State_C) that is highly opposed to nuclear proliferation would sell nuclear weapons to Terrorist_Group_T. This reduction step is rejected by the expert and the agent generates a new except-when condition and adds it to the rule. The rule refined with these except-when conditions is shown in Figure 2.

Other types of failure explanations may lead to different modifications of the rule. For instance, another strategy is to find an additional feature of the positive examples of the rule which is not a feature of the current negative example. This feature would then be added to the corresponding object from the main condition (both in the upper bound and in the lower bound).

Additional negative examples may lead to specializations of the main condition, or to generalizations of the lower bounds of existing except-when conditions, or to the learning of new except-when conditions.

The rule refinement method with a negative example is presented in more details in Table 1. As mentioned before, there are situations when several refining strategies may be applied. In the case of an example that is covered by the lower bound of the rule's main condition (without being covered by any lower bound of the except-when conditions) the agent will try first to elicit an explanation from the expert that would specialize the lower bound of the main condition, to no longer cover the negative example. If such a failure explanation is not found, the agent will try to elicit another explanation of why the example is incorrect, based on which it will learn an except-when condition.

Similarly, for a negative example that is covered by the upper bound of the rule's main condition (without being covered by the lower bound of the main condition and by the lower bound of any except-when condition) the agent will try first to specialize the upper bound of the main condition, to no longer cover the negative example. If such a specialization is not acceptable, the agent will try to elicit another explanation of why the example is incorrect, based on which it will learn a new except-when condition. In this case, the agent will also test whether this new except-when condition is more general than an existing except-when condition from the rule, in which case it will apply a lazy refinement method [3]. The agent will create a new rule in which the newly created except-when condition will replace the existing one which is less general, postponing the decision on whether to keep two separate rules or just the new version, until more examples are analyzed by the expert.

Table 1: Rule refinement with a negative example

Given:

- R – a reasoning rule
- E – a reasoning step judged by the expert as a negative example

Refine rule R to no longer cover the negative example E:

```
1. if E is covered by the upper bound of R's main condition and there is an except
   when condition's lower bound that covers E then
      keep E as a negative example for the main condition and as a positive
      example for the except-when condition
      return
2. for each except-when condition EW of the rule R do
      if E is covered by the upper bound of R's main condition and
         E is covered by EW upper bound and
         E is not covered by EW lower bound then
            attempt to generalize EW lower bound to cover E
            if any generalization performed then
               keep E as a negative example for the main condition and as a
               positive example for the except-when condition
               return
3. if E is covered by the lower bound of R's main condition and E is not covered by
   any of the except-when condition's lower bounds then
      try Main-Condition-Lower-Bound-Specialization (R, E)
      if success then keep E as a negative example and return
      else continue with step 5
4. if E is covered by the upper bound of R's main condition but not its lower
   bound, and E is not covered by any except-when condition's lower bound then
      try Main-Condition-Upper-Bound-Specialization (R, E)
      if success then keep E as a negative example and return
      else continue with step 5
5. ask for an explanation why the example E is wrong
   if an explanation is provided then
      learn an except-when condition EW-new corresponding to this explanation
      if there is another except-when condition EW such that EW-new is more
      general than EW
         create chain with a new rule obtained by replacing EW with EW-new
         return
      else add the new EW-new condition to rule R
   else keep E as a negative exception  //if no explanation is provided
```

All the above strategies refine the formal applicability condition of the rule (i.e. main condition, except-when condition). This, sometimes, requires also the refinement of the question and its answer from the informal part of the rule (see the top part of the rule in Figure 2), to better express in natural language the meaning of the formal condition. For such cases, we have developed a method which allows the expert to modify the informal description of the rule, by modifying a specific example of that rule. For instance, the expert may modify the question of the initial positive example from "Which is a nuclear state?" to "Which is a nuclear state that is not an enemy of Terrorist_Group_T and does not oppose the proliferation of nuclear weapons?" This updated question would incorporate the knowledge from the except-when conditions of the refined rule shown in Figure 2.

The modifications performed on the example are generalized and this generalization is used to update the informal structure of the rule. If the agent cannot decide whether these modifications are applicable also for previous rules examples, then a lazy rule refinement strategy is invoked [3].

4. Experimental Results

The strategies presented in this paper are integrated in the rule refinement module of Disciple, which was evaluated in two courses at the US Army War College in the past several years: “Military Applications of Artificial Intelligence” (MAAI) and “Center of Gravity Analysis” (COG) [5].

During the agent training experiments performed in the “Military Applications of Artificial Intelligence” course, 20 high-ranking military officers (13 in 2006 and 7 in 2007) trained Disciple agents to perform intelligence analysis tasks by systematically analyzing hypotheses based on relevant pieces of evidence, and experienced its use as a problem solver.

Some of the experts' assessments of the rule refinement module from the MAAI experiments performed in 2006 and 2007 are presented in Figure 5. We can see an improvement of the results in 2007, due to more strategies offered to the subject matter experts and a more natural interaction with the rule refinement tools.

The rule refinement module was also used by the knowledge engineers and subject matter experts in the initial development of the knowledge bases used in these knowledge acquisition experiments. The final knowledge base used in MAAI 2006 contained 318 reasoning rules, 140 concepts, 140 features, 260 instances, and 1400 facts. The final knowledge base for MAAI 2007 contained 360 reasoning rules, 155 concepts, 145 features, 266 instances, and 1406 facts.

The rule refinement module was also used in the development of the agent for the Center of Gravity Analysis course taught at USAWC in Spring 2007 and in the past several years. The most recent knowledge base contains 1176 reasoning rules, 357 concepts, 205 features, 226 instances, and 732 facts.

These experiments showed that the rule refinement methods can be used by the subject matter experts to refine the rules from the agent’s knowledge base to incorporate subtle distinctions in the application domain.

5. Related Research

A significant amount of work on knowledge-base refinement has been done in Machine Learning under the name of theory revision or theory refinement. Most often the problem addressed is to revise the knowledge base of a classification system to correctly classify a given set of positive and negative examples. Most of these systems correct propositional Horn-clause theories, by adding or deleting rule antecedents, and by learning new rules or deleting existing rules, to correctly classify the given set of examples (KR-FOCL [6], EITHER [7], ODYSSEUS [8], KRUST [9], MOBAL [10,11]).

The Defense Advanced Research Projects Agency (DARPA) investigated the rapid development of large knowledge bases by subject matter experts with no prior knowledge engineering experience under the Rapid Knowledge Formation (RKF) Program. Besides Disciple, two other systems for acquiring and refining expert problem solving knowledge were developed: SHAKEN and KRAKEN [12].

The SHAKEN system incorporates advanced tools that allow the subject matter experts to directly author and modify problem solving rules [13] by creating a graphical representation of a rule that can be editable by an expert through specializing, adding concepts, unifying elements, connecting elements through relations [14].

KRAKEN, which is a Cyc-based system [15] includes “knowledge engineers power tools” that can be used by subject matter experts to directly author concepts and rules. Cyc’s team is also investigating a new approach which uses machine learning techniques to automatically generate rules from ground facts [16]. After generation, these rules are reviewed by the subject matter expert and then modified by a knowledge engineer.

In the Disciple approach we ask the subject matter experts to do what they know best (i.e. to solve specific problems and

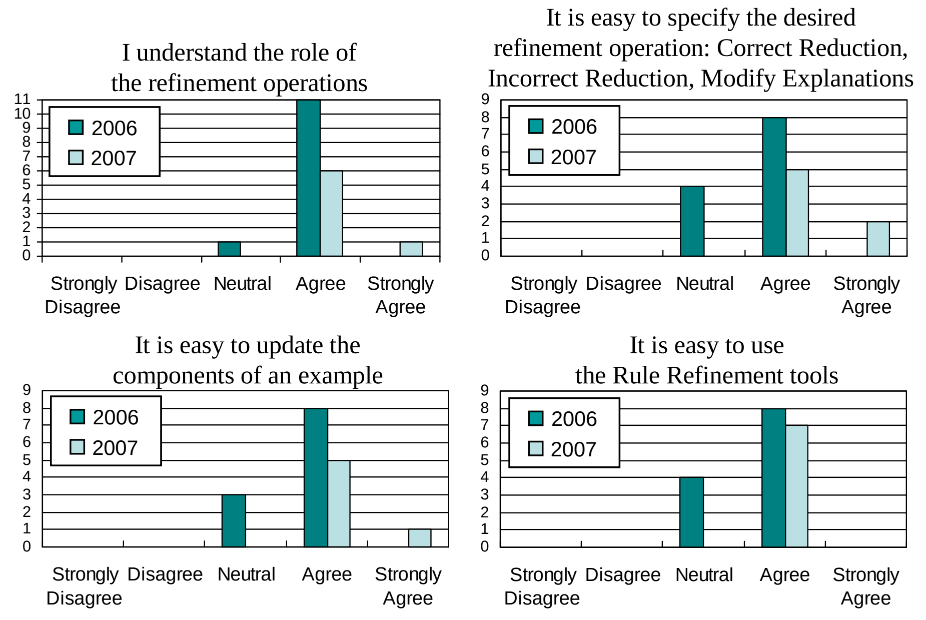

Figure 5: The experts’ assessments from the MAAI 2006 and 2007 experiments

to critique particular examples), while the agent learns and refines rules. In contrast, SHAKEN and KRAKEN attempt to facilitate direct rule authoring by the subject matter experts.

Acknowledgements

This material is based on research sponsored by several organizations, including the Air Force Research Laboratory under agreement number FA8750-04-1-0257, the Air Force Office of Scientific Research under grant number F9550-07-1-0268, the National Science Foundation under grant number 0610743 and the US Army War College. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained therein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory, Air Force Office of Scientific Research, the National Science Foundation, the US Army War College, or the U.S. Government.

References

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

[2] G. Tecuci, M. Boicu, C. Boicu, D. Marcu, B. Stanescu, M. Barbulescu. 2005. The Disciple-RKF Learning and Reasoning Agent. In *Computational Intelligence Journal (Special Issue on Learning to Improve Reasoning)*, Vol. 21, No. 4, pp. 462-479.

[3] C. Boicu. 2006. *An Integrated Approach to Rule Refinement for Instructable Knowledge-Based Agents*. PhD Dissertation in Computer Science, Learning Agents Center, Volgenau School of Information Technology and Engineering, George Mason University, Fairfax, Virginia.

[4] C. Boicu, G. Tecuci, G. and M. Boicu. 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.

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

[6] M. Pazzani and C. Brunk. 1991. Detecting and correcting errors in rule-based systems: an integration of empirical and explanation-based learning. *Knowledge Acquisition* (3), pp. 157-173.

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

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

[9] S. Craw, R. Boswell, R. Rowe. 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.

[10] K. Morik, S. Wrobel, J. Kietz, and W. Emde. 1993. *Knowledge Acquisition and Machine Learning – Theory, Methods and Applications*. London: Academic Press.

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

[12] M. Pool, K. Murray, J. Fitzgerald, M. Mehrotra, R. Schrag, J. Blythe, J. Kim, H. Chalupsky, P. Miraglia, T. Russ, D. Schneider. 2003. Evaluating Expert-Authored Rules for Military Reasoning. In *Proceedings of the 2nd International Conference on Knowledge Capture*. ACM Press, Florida.

[13] K. Barker, J. Blythe, G. Borchardt, V. Chaudhri, P.E. Clark, P. Cohen, J. Fitzgerald, K. Forbus, Y. Gil, B. Katz, J. Kim, G. King, S. Mishra, C. Morrison, K. Murray, C. Otstott, B. Porter, R.C. Schrag, T. Uribe, J. Usher, P.Z. Yeh. 2003. A Knowledge Acquisition Tool for Course of Action Analysis. In *Proceedings of the 15th Innovative Applications of Artificial Intelligence Conference*. AAAI Press, Menlo Park, California.

[14] J. Thoméré, K. Barker, V. Chaudhri, P. Clark, M. Eriksen, S. Mishra, B. Porter and A. Rodriguez. 2002. A Web-Based Ontology Browsing and Editing System. In *Proceedings of the IAAI-02*, pp. 927-934. Menlo Park, California.

[15] D. Lenat. 1995. Cyc: A Large-Scale Investment in Knowledge Infrastructure. In *Communications of the ACM*, Vol. 38, no. 11.

[16] M. Witbrock, C. Matuszek, A. Brusseau, R.C. Kahlert, C.B. Fraser, D. Lenat. 2005. Knowledge Begets Knowledge: Steps towards Assisted Knowledge Acquisition in Cyc. In *Papers from the 2005 AAAI Spring Symposium on Knowledge Collection from Volunteer Contributors* (KCVC), pp. 99-105. Stanford, California.