Ontologies for Learning Agents: Problems, Solutions and Directions

Bogdan Stanescu, Cristina Boicu, Gabriel Balan, Marcel Barbulescu, Mihai Boicu, Gheorghe Tecuci
4A5, Learning Agents Laboratory, George Mason University
4400 University Dr., Fairfax, VA-22030, US
{bstanesc, ccascava, gbalan, mbarbule, mboicu, tecuci}@gmu.edu

Abstract

We are developing a general end-to-end approach, called Disciple, for building and using personal problem solving and learning agents. This approach raises complex challenges related to ontology specification, import, elicitation, learning, and merging, that we have explored to various degrees, as we are developing successive versions of Disciple. This paper presents some of these challenges, our current solutions and the future directions, that are relevant for building agents in general.

1 Introduction

The long term objective of our research is to develop the science and technology that will allow typical computer users to train and use their personal intelligent assistants. Our approach to this problem is to develop a series of increasingly more capable agents from the Disciple family of learning agent shells [Tecuci, 1998; Tecuci *et al.*, 2002]. A Disciple agent can be initially trained by a subject matter expert and a knowledge engineer, in a way that is similar to how an expert would teach an apprentice, through problem solving examples and explanations. Once trained to a significant level of competence, copies of the agent are handed over to typical computer users. These agents then assist their users through mixed-initiative reasoning, increasing their recall, speed and accuracy, without impeding their creativity and flexibility. In the same time, the assistants continue to learn from this joint problem solving experience, adapting to their users to become better collaborators that are aware of users' preferences, biases and assumptions.

The process of building and using such problem solving and learning agents raises complex challenges related to ontology specification, import, elicitation, learning, and merging, that we have explored to various degrees, as we are developing successive versions of Disciple. The goal of this paper is to present some of these challenges, our current solutions and the future directions, that are relevant for building agents in general.

In the last three years, the development of the Disciple approach was driven by the attempt to find an automatic solution to the complex Center of Gravity (COG) analysis problem, in collaboration with the US Army War College. The center of gravity of a force (state, alliance, coalition or group) represents the foundation of capability, power and movement, upon which everything depends [Clausewitz, 1976]. In any conflict, a force should concentrate its effort on its enemy's center of gravity, while adequately protecting its own. As a consequence, the examples used in this paper will be from the COG domain, but they will not require an understanding of this domain.

The rest of this paper is organized as follows. The next section discusses the use of the ontology for representation, communication, problem solving, and learning, both in general, and in the context of the Disciple family. Section 3 gives an overview of the Disciple agent building methodology, stressing the ontology-related activities. Then sections 4 to 7 discuss in more details some of our main results on ontology specification, exception-based ontology learning, example-based ontology learning, and ontology import and merging. These sections will include experimental results and plans for future research.

2 Knowledge representation for problem solving and learning

A Disciple learning agent shell includes general problem-solving and learning engines for building a knowledge base consisting of an object ontology that specifies the terms from a particular domain, and a set of problem solving rules expressed with these terms [Tecuci *et al.*, 2002]. The problem-solving engine is based on the general task reduction paradigm. In this paradigm, a task to be performed is successively reduced to simpler tasks, by applying task reduction rules. Then the solutions of the simplest tasks are successively combined, by applying solution composition rules, until they produce the solution of the initial task.

The object ontology is a hierarchical representation of the objects and types of objects from the application domain. It represents the different kinds of objects, the properties of each object, and the relationships existing between objects. A fragment of the object ontology for the COG domain is shown in the bottom part of Figure 1.

The reduction rules are IF-THEN structures that express how and under what conditions a certain type of task may be reduced to simpler subtasks. The reduction

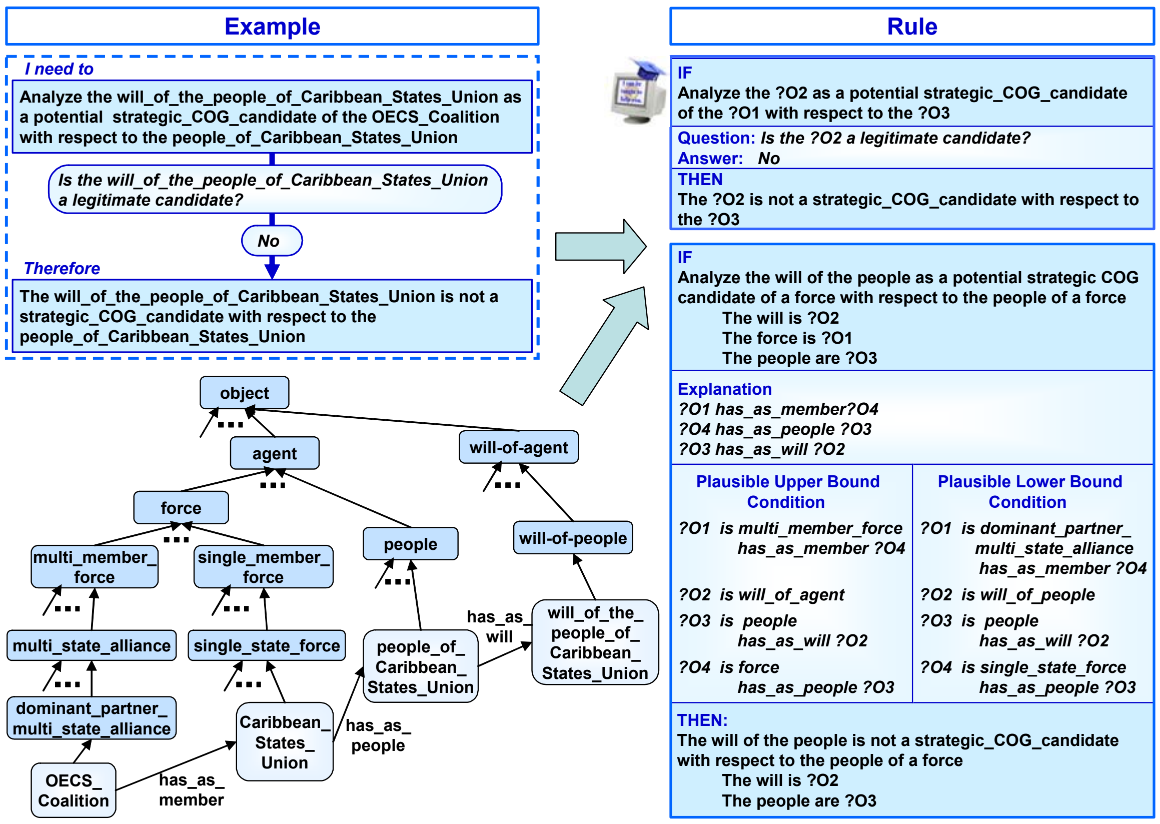

Example
I need to
Analyze the will_of_the_people_of_Caribbean_States_Union as a potential strategic_COG_candidate of the OECS_Coalition with respect to the people_of_Caribbean_States_Union
Is the will_of_the_people_of_Caribbean_States_Union a legitimate candidate?
No
Therefore
The will_of_the_people_of_Caribbean_States_Union is not a strategic_COG_candidate with respect to the people_of_Caribbean_States_Union
object
agent
will-of-agent
force
people
will-of-people
multi_member_force
single_member_force
multi_state_alliance
single_state_force
people_of_Caribbean_States_Union
will_of_the_people_of_Caribbean_States_Union
has_as_will
dominant_partner_multi_state_alliance
Caribbean_States_Union
has_as_people
OECS_Coalition
has_as_member
Rule
IF
Analyze the ?O2 as a potential strategic_COG_candidate of the ?O1 with respect to the ?O3
Question: Is the ?O2 a legitimate candidate?
Answer: No
THEN
The ?O2 is not a strategic_COG_candidate with respect to the ?O3
IF
Analyze the will of the people as a potential strategic COG candidate of a force with respect to the people of a force
The will is ?O2
The force is ?O1
The people are ?O3
Explanation
?O1 has_as_member?O4
?O4 has_as_people ?O3
?O3 has_as_will ?O2
Plausible Upper Bound Condition
?O1 is multi_member_force has_as_member ?O4
?O2 is will_of_agent
?O3 is people has_as_will ?O2
?O4 is force has_as_people ?O3
Plausible Lower Bound Condition
?O1 is dominant_partner_multi_state_alliance has_as_member ?O4
?O2 is will_of_people
?O3 is people has_as_will ?O2
?O4 is single_state_force has_as_people ?O3
THEN:
The will of the people is not a strategic_COG_candidate with respect to the people of a force
The will is ?O2
The people are ?O3

Figure 1: Ontology based rule learning.

rule are paired with IF-THEN composition rules that express how and under what conditions the solutions of the subtasks may be composed into the solution of the task. An example of a simple task reduction rule is shown in the right hand side of Figure 1. In this case the IF task is reduced to its solution.

The learning engines use several strategies to learn the rules and to refine the object ontology. At the basis of the learning methods are the notion of plausible version space [Tecuci, 1998; Boicu, 2002] and the use of the object ontology as an incomplete and partially incorrect generalization hierarchy for learning.

A plausible version space is an approximate representation for a partially learned concept, as illustrated in Figure 2. The partially learned concept is represented by a plausible upper bound concept which, as an approximation, is more general than the concept E_h to be learned, and by a plausible lower bound concept which, again as an approximation, is less general than E_h. During learning, the two bounds (which are first order logical expressions) converge toward one another through successive generalizations and specializations, approximating E_h better and better.

The partially learned knowledge pieces from the knowledge base of Disciple are represented with plausible version spaces. Notice, for example, that the IF-THEN rule from the bottom right part of Figure 1 does not have a single applicability condition but two conditions (Plausible Lower Bound Condition and Plausible Upper Bound Condition) that define the plausible version space of the exact condition of the rule. Similarly, each partially learned feature F from the object ontology has its domain and range represented as plausible version spaces. The domain to be learned of the feature F is a concept that represents the set of objects that could have the feature F. Similarly, the range to be learned is a concept that represents the set of possible values of F.

The object ontology plays a crucial role in Disciple, being at the basis of user-agent communication, problem solving, knowledge acquisition and learning. First of all, the object ontology provides the basic representational constituents for all the elements of the knowledge base. When an expert teaches a Disciple agent, the expert expresses his/her reasoning process in natural language, as illustrated by the task reduction example in the upper left side of Figure 1. The top task is the task to be reduced. In order to reduce this task the expert asks a relevant

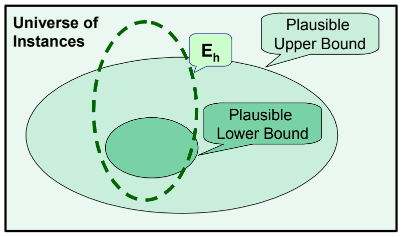


Figure 2: A representation of a plausible version space

question. The answer to this question leads to the reduction of this task to a solution. As the expert types these expressions using natural language, the agent interacts with him/her to replace certain phrases with the ontology terms they designate (e.g. “will of the people of Caribbean State Union” or “strategic COG candidate”). The recognition of these terms facilitates the understanding of the expert’s phrases and the learning of a general rule from this specific example. The learned rule has an informal structure (shown in the top right part of Figure 1) and a formal structure (shown in the bottom right part of Figure 1). The informal structure preserves the natural language of the expert and is used in agent-user communication. The formal structure is used in the actual reasoning of the agent. Notice that the two plausible version space conditions from the formal structure are expressed with the terms from the object ontology. The formal tasks and their features are also part of the task ontology, and feature ontology, respectively.

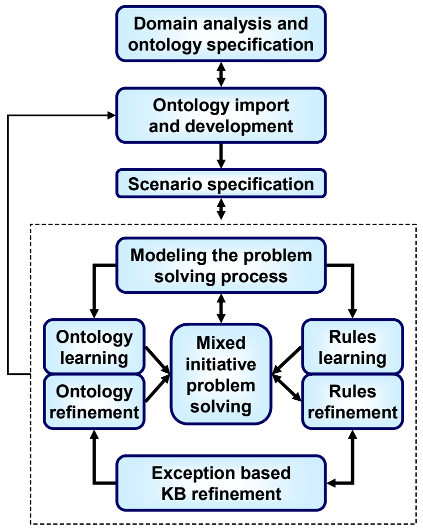
As mentioned above, the object ontology has a fundamental role in learning, being used as a generalization hierarchy. Indeed, notice that the specific instances from the example (“will of the people of Caribbean State Union”, “OECS Coalition”, “people of Caribbean State Union”) are replaced in the learned rule with more general concepts from the object ontology (“will of agent”, “multi member force”, “people”), and their relationships.

While the corresponding learning algorithm is presented in [Boicu *et al.*, 2000; Boicu 2002], it is important to stress here that the agent’s generalization hierarchy (the object ontology) is itself evolving during learning (as discussed in sections 4, 5, and 6). Therefore Disciple addresses the complex and more realistic problem of learning in the context of an evolving representation language. The next section gives an overview of the agent building methodology, stressing the ontology-related activities.

3 Agent building methodology

The Disciple learning agent shell could be used to rapidly develop a Disciple agent for a specific application domain, by following the steps from Figure 3. There are two main phases in this process: the development of an initial object ontology and the teaching of the agent. The first phase has to be performed jointly by a knowledge engineer and a subject matter expert. The second phase may be performed primarily by the subject matter expert, with limited assistance from a knowledge engineer.

During *domain analysis and ontology specification*, the knowledge engineer works with the subject matter expert to develop an initial model of how the expert solves problems, based on the task reduction paradigm. The model identifies also the object concepts that need to be represented in Disciple’s ontology so that it can perform this type of reasoning. These object concepts represent a specification of the ontology needed for reasoning.


Domain analysis and ontology specification
Ontology import and development
Scenario specification
Modeling the problem solving process
Ontology learning
Ontology refinement
Mixed initiative problem solving
Rules learning
Rules refinement
Exception based KB refinement
Figure 3: Main agent development processes.

During *ontology import and development*, this specification guides the process of importing ontological knowledge from existing knowledge repositories, such as CYC [Lenat, 1995], as discussed in section 7. However, not all the necessary terms will be found in external repositories and therefore the knowledge engineer and the subject matter expert will also have to extend the imported ontology using the ontology development tools of Disciple. For instance, Figure 4 shows the interfaces of three different ontology browsers of Disciple, the association browser (which displays and objects and its relationships with other objects), the tree browser (which displays the hierarchical relationships between the objects in a tree structure), and the graphical browser (which displays the hierarchical relationships between the objects in a graph structure).

Once the object ontology is developed, the knowledge engineer has to define elicitation scripts using the Script Editor of Disciple. The elicitation scripts will be executed by the Scenario Elicitation tool, guiding the user of Disciple to define a specific scenario or problem solving situation (e.g. the current war on terror, including the characteristics of the participating forces, such as US and Al Qaeda). This process will be described in more detail in section 4. The result of this initial KB development phase is an object ontology with instances characterizing a specific scenario.

In the next major phase, the subject matter expert will use the current scenario to teach Disciple how to solve problems (e.g. how to determine the centers of gravity of the opposing forces in the current war on terror).

First, the expert will interact with the Modeling advisor tool of Disciple. This tool will assist the expert to express his or her reasoning process in English, using the task reduction paradigm. The result of this process will be task reduction steps like the one from the upper left part of Figure 1. These steps may also include new terms that are not yet present in the object ontology of Disciple. Each such term is an example for learning a general con-

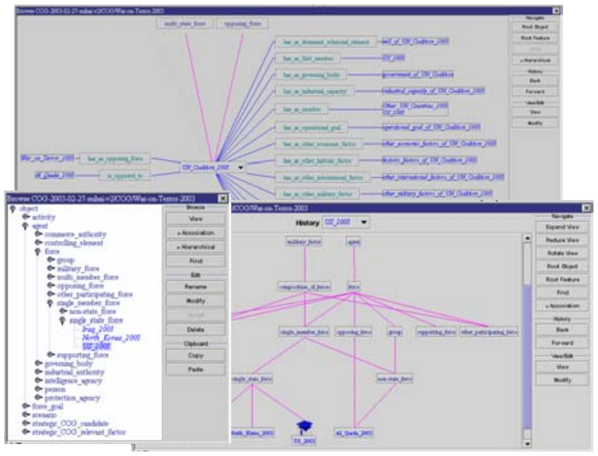


Figure 4: Association, tree, and hierarchical browsers.

cept or a general feature using the Ontology learning method discussed in section 6. Also, each specific reasoning step formulated with the Modeling advisor is an example from which a general rule is learned using the Rule Learning tool. An example of such a rule is presented in the right hand side of Figure 1.

As Disciple learns more rules, the interaction with the subject matter experts evolves from a teacher-student type of interaction to an interaction where both collaborate in solving a problem. This interaction is governed by the mixed-initiative problem solving tool. In this case, Disciple uses the partially learned rules to propose solutions to the current problems, and the expert's feedback will be used by the Rule Refinement tool and the Ontology Refinement tool to improve both the rules and the ontology elements involved in the rules' applications.

There is no fixed sequence of tool invocations. Instead, they are used opportunistically, based on the current problem solving situation. For example, while the expert and Disciple are performing mixed-initiative problem solving, the expert may need to define a new reduction that requires modeling, rule learning and rule refinement.

Because the rule learning and refinement processes take place in the context of an incomplete and partially incorrect object ontology, some of the learned rules may accumulate exceptions. In such a case, the exception-based KB refinement tool may be invoked to extend or correct the object ontology and to correspondingly refine the rules. This process will be presented in section 5.

Because one of the goals of this research is the rapid development of knowledge bases, the Disciple shell also includes tools to merge the ontologies and the rules developed in parallel by the subject matter experts. Section 7 discusses this issue in more detail.

In the last three years we have performed extensive experiments with Disciple at the US Army War College, where it is used in two courses, Case Studies in Center of Gravity Analysis (the COG course), and Military Applications of Artificial Intelligence (the MAAI course). In the COG course, Disciple is used as an assistant that was trained by the instructor, helping the students to perform a COG analysis of a scenario and to generate an analysis report. Over 95% of the students from the 2002 Terms II and III sessions of this course agreed with the following statement: *Disciple helped me to learn how to perform a strategic center of gravity analysis of a scenario*. In the follow-on MAAI course, the students taught personal Disciple agents their own expertise in COG analysis. After the experiments conducted in Spring 2001 and Spring 2002, 19 of the 25 students agreed (and 6 were neutral) with the statement: *I think that a subject matter expert can use Disciple to build an agent, with limited assistance from a knowledge engineer*.

The following sections will provide more details on some of the most important ontology-related processes of the Disciple agent development methodology, as well as results from the above experiments.

4 Scenario specification

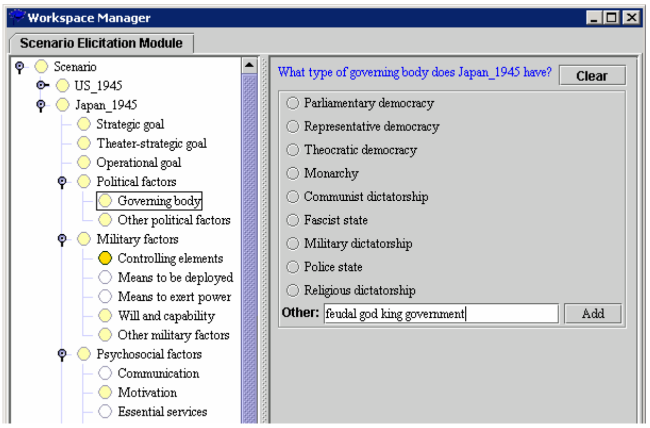
As part of the initial ontology development, the knowledge engineer uses the Script Editor to define elicitation scripts that specify how to elicit the description of a scenario from the user. These scripts are associated with the concepts and features from the ontology. Each script has a name, a list of arguments, and it specifies how to display the dialog with the user, the questions to ask the user, how to store the answers in the ontology, and what other scripts to call. Table 1 shows the script "elicit government type" associated with the concept "state government".

The elicitation scripts are executed by the Scenario Elicitation tool. As illustrated in Figure 5, the left hand side of the Scenario Elicitation interface displays a table of contents. When the expert clicks on one of these titles, questions that elicit the corresponding description are displayed in the right hand side of the screen. The use of the elicitation scripts allows a knowledge engineer to rapidly build a customized interface for a Disciple agent, thus effectively transforming this software development task into a knowledge engineering one.

The Protégé system [Noy *et al.,* 2000] has a similar capability of using elicitation scripts to acquire instances of concepts. However, Disciple extends Protégé in several directions. In Disciple the expert does not need to see or understand the object ontology in order to answer the questions and describe a scenario. Instead, the expert-agent interaction is directed by the execution of the scripts. Once the expert answers some questions or up-

Table 1: Sample elicitation script.

```
Script: state_government.elicit government type
Arguments: <force-name>, <government-name>
Control: single-selection-list
    Question: What type of government does <force-name> have?
    Answer variable: <government-type>
    Possible values: the elementary subconcepts of state_government
    Allow adding new subconcepts: Yes
Ontology actions:
    <government-name> instance-of <government-type>
Script call: <government-type>.elicit properties
    Arguments: <government-name>
```



What type of governing body does Japan_1945 have?

Other: feudal god king government

Figure 5: Execution of the elicitation script from Table 1.

dates his answers, new titles may be inserted into the table of contents, as directed by the script calls. For instance, after the expert specifies the opposing forces in a scenario, their names appear as titles in the table of contents, together with the characteristics that need to be elicited for them. Experimental results show that the experts can easily use the Scenario Elicitation module [Tecuci *et al.*, 2002].In Protégé, each concept has exactly one script that specifies how to elicit the properties of its instances. In Disciple, a concept can have any number of scripts that can be used for any purpose. In particular, the knowledge engineer can define more scripts that specify how to elicit instances for the same concept. For instance, to elicit the military factors for a single-state force, different questions have to be asked if the force is part of an alliance, or is a standalone opposing force.

The most recent development of the Scenario Elicitation tool is to allow the user to extend the ontology with new concepts in a controlled manner. For instance when the script from Table 1 is executed, the user can specify a new type of state government (e.g. "feudal god-king government"), as illustrated in Figure 5. As a result a new concept is created under "state government". As future developments, we plan to extend the capability of the Script Editor to facilitate the script definition task for the knowledge engineer, by taking into account the structure of the ontology and by using customization of generic scripts. We also plan to add natural language processing capabilities to the Scenario Elicitation module.

5 Exception-based ontology learning

As we have mentioned in section 2, the object ontology plays a crucial role in the learning process of the agent, as it is used as the generalization hierarchy for learning. However, this ontology is itself incomplete and partially incorrect and will have to be improved during the teaching of the agent. In this section we will briefly present an exception-based approach to ontology learning.

Because the ontology is incomplete, it may not contain the knowledge to distinguish between all the positive examples and the negative examples of a learned rule, such as the one presented in Figure 1. As a result, a rule may accumulate negative and positive exceptions.

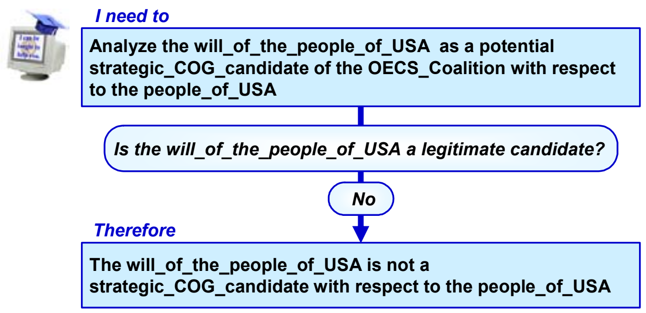
A *negative exception* is a negative example that is covered by the rule because the current object ontology does not contain any knowledge that distinguishes the negative example from the positive examples of the rule [Tecuci, 1998; Boicu *et al.*, 2003]. Therefore, the rule cannot be further specialized to uncover the negative example, while still covering all the positive examples of the rules. A *positive exception* is defined in a similar way.

A comparative analysis of the examples and the exceptions will facilitate identifying what distinguishes them and how the object ontology needs to be extended to incorporate the identified distinction. This is precisely the main idea behind our exception-based learning method in which a subject matter expert collaborates closely with the agent to discover possible ontology extensions (such as new concepts, new features or new feature values) that will eliminate the exceptions.

The exception-based learning method consists of four main phases: 1) a *candidate discovery* phase in which the agent analyzes a rule, its examples and exceptions, and the ontology and finds the most plausible types of extensions of the ontology that may reduce or eliminate the rule's exceptions; 2) a *candidate selection* phase in which the expert interacts with the agent to select one of the proposed candidates; 3) an *ontology refinement* phase in which the agent elicits the ontology extension knowledge from the expert and 4) a *rule refinement* phase in which the agent updates the rule and eliminates the rule's exceptions based on the performed ontology extension.

As an illustration, consider the example and the corresponding partially learned rule from Figure 1. This rule is used in problem solving and generates the reasoning step from Figure 6, which is rejected by the expert because both the answer to the question and the resulting solution are wrong. However, there is no knowledge in the current ontology that can distinguish between the objects from the positive example in Figure 1 and the corresponding objects from the negative example in Figure 6. Therefore, the negative example from Figure 6 will be kept as a negative exception of the rule in Figure 1.

Figure 7 shows the interface of the exception-based learning tool in the ontology refinement phase. The upper left panel of this tool shows the negative exception which needs to be eliminated. Below are the objects that are currently differentiated: "Caribbean States Union" (from the positive example) and "USA" (from the negative exception). The right panel shows the elicitation dialog, in which the expert is guided by the agent to indicate the name and value of a new feature that expresses the difference between "Caribbean States Union" and "USA." The expert defines the new feature "is minor member of" and specifies that "Caribbean States Union" *is a minor member of* "OECS Coalition," while "USA" is not. Based on this elicitation, Disciple learns a general definition of the feature "*is minor member of*" and refines the ontology to incorporate this knowledge. A fragment of the refined


I need to

Analyze the will_of_the_people_of_USA as a potential strategic_COG_candidate of the OECS_Coalition with respect to the people_of_USA

Is the will_of_the_people_of_USA a legitimate candidate?

No

Therefore

The will_of_the_people_of_USA is not a strategic_COG_candidate with respect to the people_of_USA

Figure 6: Incorrect reasoning step generated by the agent

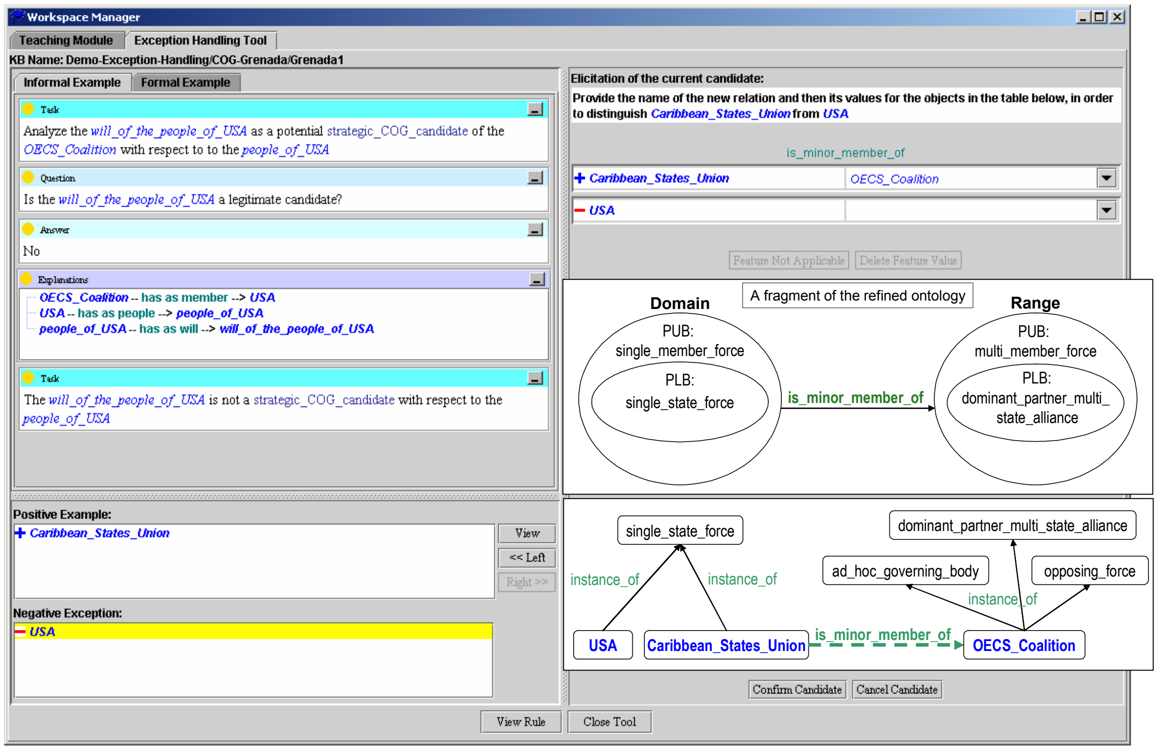
ontology is shown in the right part of Figure 7. Notice that both the domain and the range of the new feature are represented as plausible version spaces. The plausible upper bound domain of this feature is "single member force" and the plausible lower bound domain is "single state force."

The exception-based learning tool was evaluated during the Spring 2002 agent teaching experiment performed with Disciple at the US Army War College, as part of the “Military Applications of Artificial Intelligence” course. The tool was used by seven subject matter experts with the assistance of a knowledge engineer, to eliminate the negative exceptions of the rules. We did not expect a significant number of exceptions, because before the experiment we attempted to develop a complete ontology, which contained 191 concepts and 206 features. However, during the experiment, 8 of the learned problem solving rules have collected 11 negative exceptions, indicating that the ontology was not complete. In order to eliminate these exceptions, the experts extended the ontology with 4 new features and 6 new facts. Some of the newly created features eliminated the exceptions from several rules. As a result of these ontology extensions, the rules were correspondingly refined.

This experiment proved that the exception-based learning tool can be used to extend the object ontology with new elements that represent better the subtle distinctions that the experts make in their domains of expertise. This tool allows the elimination of the rules' exceptions and it improves the accuracy of the learned rules by refining their plausible version space conditions. It also enhances the agent's problem solving efficiency by eliminating the need to explicitly check the exceptions. We plan several extensions to the presented method: propose suggestions and help the user during the exception-based learning process; use analogical reasoning and hints from the user in the discovery of plausible ontology extensions; extend the method to discover new object concepts in order to eliminate the rules' exceptions; and extend the method to also remove the positive exceptions of the rules.

6 Example-based ontology learning

There are many situations during the agent teaching process where the subject matter expert has to specify a fact involving a new instance or a new feature. In such a case, the example-based ontology learning tool is invoked to learn a new concept or a new feature definition, from the provided fact. One such situation was encountered in the previous section where the expert indicated that “Caribbean States Union *is minor member of* OECS



KB Name: Demo-Exception-Handling/COG-Grenada/Grenada1

Task: Analyze the *will_of_the_people_of_USA* as a potential strategic_COG_candidate of the *OECS_Coalition* with respect to to the *people_of_USA*

Question: Is the *will_of_the_people_of_USA* a legitimate candidate?

Answer: No

Explanations:
OECS_Coalition -- has as member --> *USA*
USA -- has as people --> *people_of_USA*
people_of_USA -- has as will --> *will_of_the_people_of_USA*

Task: The *will_of_the_people_of_USA* is not a strategic_COG_candidate with respect to the *people_of_USA*

Positive Example: + *Caribbean_States_Union*

Negative Exception: − *USA*

Elicitation of the current candidate:
Provide the name of the new relation and then its values for the objects in the table below, in order to distinguish *Caribbean_States_Union* from *USA*

is_minor_member_of
+ *Caribbean_States_Union* — *OECS_Coalition*
− *USA* —

Figure 7: The interface of the Exception-Based Learning Module and a fragment of the refined ontology

Coalition." From this specific fact Disciple attempts to learn a general definition of the feature "*is minor member of*." The most important characteristics of the feature that need to be learned are its position in the feature hierarchy, its domain of applicability, and its range of possible values. First Disciple identifies the features that are most likely to be more general than *"is minor member of."* This set initially includes all the features whose domain and range cover "Caribbean States Union" and "OECS Coalition," respectively, as shown in Figure 8. This set if further pruned by applying various heuristics (for instance by eliminating the other features of "Caribbean States Union") and by directly asking the expert:

Consider the statement "Caribbean States Union *is minor member of* OECS Coalition." Is this a more specific way of saying: "Caribbean States Union *is member of* OECS Coalition"?

As a result of this process "*is minor member of*" is defined as a subfeature of "*is member of*." The domain and the range of the "*is member of*" feature become the upper bounds of the domain and range of "*is minor member of*." The corresponding lower bounds are the minimal generalizations of "Caribbean States Union" and "OECS Coalition," respectively (see the bottom part of Figure 7).

The next step is to further refine the plausible version spaces of the domain and range. The lower bounds are generalized based on new positive examples of this feature, encountered during further teaching. However, the agent will not encounter negative examples. Therefore the specialization of the upper bounds is based on a dialog with the expert who will be asked to identify objects that cannot have this feature, or cannot be a value of this feature. There are other difficult problems related to learning and refining features: how to elicit its special characteristics (e.g. whether the feature is transitive or not), how to elicit its cardinality, or how to differentiate between required and optional features for an object.

7 Ontology import and merging

Figure 9 shows another view of the Disciple agent building methodology that emphasizes ontology reuse and parallel knowledge base development. The ontology specification that results from the domain analysis phase (see Figure 3) guides the process of importing ontological knowledge, currently from CYC [Lenat, 1995] and, in the future, also from other knowledge repositories.

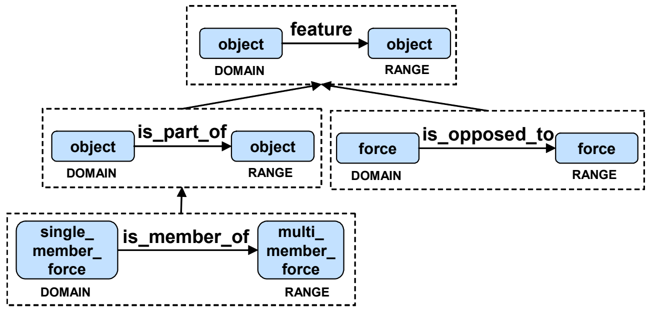


Figure 8: Fragment of the feature hierarchy.

Our import method consists of identifying key terms in the CYC KB that correspond to the terms from the ontology specification, extracting the knowledge related to those terms and importing it into the Disciple knowledge base. The extraction of knowledge is an automated process in which all the terms related to the start-up terms are elicited, then all the terms related to those terms, and so on until a transitive closure or a user-specified stopping criteria is met. This method extends the one of Chaudhri *et al.* [2000] by adding stopping criteria, by allowing taxonomy relations to be followed down the hierarchy, and by considering the feature hierarchy. The translation of the extracted knowledge into the Disciple formalism consists of a syntactic phase and a semantic one, being similar with the method used in *OntoMorph* [Chalupsky, 2000]. During the automatic transformation of extracted knowledge into Disciple's knowledge representation, the system records logs with a number of decisions that require the user's approval or refinement.

The imported ontology is further extended using the ontology development tools of Disciple, as discussed in section 3, leading to an initial knowledge denoted with KB0 in Figure 9.

Another result of the Domain analysis phase is a partitioning of the application domain into several subdomains. A team of experts can now develop separate knowledge bases for each independent subdomain. Each expert teaches a personal Disciple agent, starting from the common knowledge base KB0 and building a refined one, as indicated in Figure 9. Then, the developed knowledge bases are merged into the Final KB. This KB will contain a merged ontology, but separate partitions of rules, one for each subdomain. The ontology merging algorithm exploits the fact that the KBs to be merged share KB0 as a common ontology. It starts with one of the KBs and successively merges it with the other KBs,

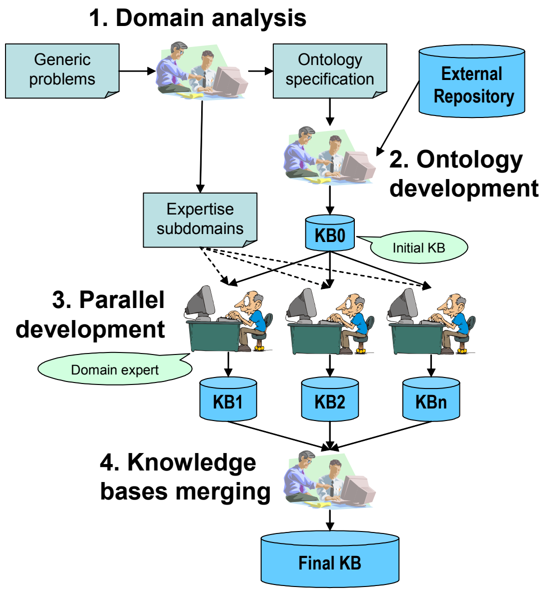


Figure 9: Rapid knowledge base development.

one at a time. Similarly to *Prompt* [Noy and Musen, 2000] and *Chimaera* [McGuiness *et al.*, 2000], our approach to merging is based on providing an interactive way of copying one frame from an ontology into the other. While it is acknowledged that the role of the human cannot be eliminated from this process [Klein, 2001; Noy and Musen, 2000], the goal is to provide the most assistance to the knowledge engineer. Therefore, our tool handles the low level operations, allowing the user to issue only the most general commands, and assuring that the ontology is kept consistent at all times. In addition to that, the agent makes suggestions and keeps the user focused on the part of the ontology being merged.

The parallel KB development and merging capabilities of Disciple were first evaluated in Spring 2002, as part of "IT 803 Intelligent Agents" course at George Mason University. The students had to develop an agent for helping someone to choose a PhD advisor. The domain was split into six parts that were developed separately by the students in the class. They started the knowledge base development with a general 23-fact knowledge base provided by the instructor and each of them had to extend it with the knowledge needed to express their own part of the domain. Each student extended its knowledge base with an average of 97 facts. Using the merging tools provided by Disciple, the students succeeded to merge all their work into a single agent with an ontology containing 473 facts. We plan to validate the entire methodology in a new experiment at the US Army War College, as part of the Spring 2003 MAAI course.

Future work includes the capability to import from OKBC knowledge servers [Chaudhri *et al.*, 1998] and from DAML+OIL expressed ontologies [Connolly *et al.*, 2001], and an improvement of the proactivity of the mixed-initiative ontology merging tool.

Acknowledgements. This research was sponsored by DARPA, AFRL, AFMC, USAF, under agreement number F30602-00-2-0546, by the AFOSR under grant no. F49620-00-1-0072, and by the US Army War College.

References

[Boicu *et al.*, 2003] Cristina Boicu, Gheorghe Tecuci, Mihai Boicu, and Dorin Marcu. Improving the Representation Space through Exception-Based Learning. To appear in *Proceedings of the Sixteenth International Flairs Conference.* 2003.

[Boicu *et al.,* 2000] Mihai Boicu, Gheorghe Tecuci, Dorin Marcu, Michael Bowman, Ping Shyr, Florin Ciucu, and Cristian Levcovici. Disciple-COA: From Agent Programming to Agent Teaching. In *Proceedings of the Seventeenth International Conference on Machine Learning,* Stanford, California, 2000. Morgan Kaufmann.

[Boicu, 2002] Mihai Boicu. *Modeling and Learning with Incomplete Knowledge*. Doctoral Dissertation. George Mason University, Fairfax, Virginia, 2002.

[Chalupsky, 2000] Hans Chalupsky. OntoMorph: a translation system for symbolic knowledge. In *Proceedings of Seventh International Conference on Knowledge Representation and Reasoning,* pages 471--482, San Francisco, California, April 2000. Morgan Kaufmann.

[Chaudhri *et al.*, 1998] Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, and James P. Rice. OKBC: A Programmatic Foundation for Knowledge Base Interoperability. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 600--607, Madison, Wisconsin, July 1998. AAAI Press/The MIT Press.

[Chaudhri *et al.*, 2000] Vinay K. Chaudhri, Mark E. Stickel, Jerome F. Thomere, and Richard J. Waldinger. Using Prior Knowledge: Problems and Solutions. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 436--442. Austin, Texas, July-August 2000. AAAI Press/The MIT Press.

[Clausewitz, 1976] Clausewitz, C.V.. *On War*. Translated and edited by Howard, M. and Paret, P. Princeton University Press, Princeton, NJ.

[Connolly *et al.*, 2001] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. DAML+OIL (March 2001) Reference Description. W3C Note 18 December, 2001.

[Noy *et al.,* 2000] Natalya Fridman Noy, Ray W. Fergerson, and Mark A. Musen. The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility. In *Proceedings of the European Knowledge Acquisition Workshop*, pages 17-32, 2000.

[Klein, 2001] Michel Klein. Combining and relating ontologies: an analysis of problems and solutions. In *Proceedings of the IJCAI-20001 Workshop on Ontologies and Information Sharing*, Seattle, Washington, August 2001. International Joint Conference on Artificial Intelligence, Inc.

[Lenat, 1995] Douglas B. Lenat. CYC: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM*, 38(11): 33-38, 1995.

[McGuinness *et al.*, 2000] Deborah L. McGuinness, Richard E. Fikes, James Rice, and Steve Wilder. An Environment for Merging and Testing Large Ontologies. In *Proceedings of Seventh International Conference on Knowledge Representation and Reasoning*, San Francisco, California, April 2000. Morgan Kaufmann.

[Noy and Musen, 2000] Natalya F. Noy and Mark A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 450–455, Austin, Texas, July–August 2000. AAAI Press/The MIT Press.

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

[Tecuci *et al.*, 2002] Gheorghe Tecuci, Mihai Boicu, Dorin Marcu, Bogdan Stanescu, Cristina Boicu, and Jerome Comello. Training and Using Disciple Agents: A Case Study in the Military Center of Gravity Analysis Domain. *AI Magazine*, 23(4): 51—68, 2002.