REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 28-02-2010	**2. REPORT TYPE** Final Report	**3. DATES COVERED** *(From - To)* From 04-04-2007 To 30-11-2009

4. TITLE AND SUBTITLE Agent Learning for Mixed-Initiative Knowledge Acquisition	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER FA9550-07-1-0268
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Tecuci Gheorghe, Boicu Mihai	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Learning Agents Center, MSN 6B3, George Mason University, 4400 University Drive, Fairfax VA 22030-4444; phone: (703) 993 1722, fax: (703) 993 9275, email: tecuci@gmu.edu, http://lac.gmu.edu	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFOSR Grant FA9550-07-1-0268
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Scientific Research, 875 North Randolph Street, Suite 325, Rm 3112, Arlington, VA 22203	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFOSR
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution of this document is unlimited

13. SUPPLEMENTARY NOTES

AFOSR Program Manager: Dr. David R. Luginbuhl

14. ABSTRACT This research has advanced the Disciple learning and problem solving theory for rapid development and maintenance of adaptable cognitive assistants in uncertain and dynamic environments. These assistants can capture, use, preserve, and transfer to other users the subject matter expertise which currently takes years to establish, is lost when experts separate from service, and is costly to replace. The research has developed an integrated set of methods for representing partially learned knowledge, for knowledge acquisition through teaching and learning, for evidence-based reasoning with Wigmorean inference networks integrating logic and probabilities, and for mixed-initiative interaction. These methods have been implemented into a general agent shell that incorporates a large amount of knowledge for evidence-based reasoning from the Science of Evidence. The agent shell can be taught to solve problems by a subject matter expert, with assistance from a knowledge engineer, to become a domain-specific cognitive assistant. The shell has been used to develop several prototype cognitive assistants. For instance, the cognitive assistant for intelligence analysis demonstrated mixed-initiative learning of complex analysis rules and mixed-initiative hypothesis analysis, allowing the analyst to drill-down based on available time and evidence, to make probabilistic assumptions, and to revise the analysis in light of new evidence.

15. SUBJECT TERMS agent learning, knowledge acquisition, evidence-based reasoning, Wigmorean networks, Science of Evidence, mixed-initiative interaction, problem reduction and solution synthesis, ontology, rules, cognitive assistant, subject matter expertise, knowledge engineering, intelligence analysis

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT unclassified	**b. ABSTRACT** unclassified	**c. THIS PAGE** unclassified	unclassified unlimited	36	**19b. TELEPHONE NUMBER** *(include area code)*

AGENT LEARNING FOR MIXED-INITIATIVE KNOWLEDGE ACQUISITION

Gheorghe Tecuci and Mihai Boicu

Learning Agents Center, George Mason University, Fairfax, VA 22030

Abstract: This research has advanced the Disciple learning and problem solving theory for rapid development and maintenance of adaptable cognitive assistants in uncertain and dynamic environments. These assistants can capture, use, preserve, and transfer to other users the subject matter expertise which currently takes years to establish, is lost when experts separate from service, and is costly to replace. The research has developed an integrated set of methods for representing partially learned knowledge, for knowledge acquisition through teaching and learning, for evidence-based reasoning with Wigmorean inference networks integrating logic and probabilities, and for mixed-initiative interaction. These methods have been implemented into a general agent shell that incorporates a large amount of knowledge for evidence-based reasoning from the Science of Evidence. The agent shell can be taught to solve problems by a subject matter expert, with assistance from a knowledge engineer, to become a domain-specific cognitive assistant. The shell has been used to develop several prototype cognitive assistants. For instance, the cognitive assistant for intelligence analysis demonstrated mixed-initiative learning of complex analysis rules and mixed-initiative hypothesis analysis, allowing the analyst to drill-down based on available time and evidence, to make probabilistic assumptions, and to revise the analysis in light of new evidence.

Acknowledgements: We are gratefully to the Air Force Office of Scientific Research and to David Luginbuhl, Robert Herklotz, and Alex Kilpatrick for the support provided over the years to the development of the Disciple learning and problem solving theory. We are also grateful to David A. Schum for our very fruitful collaboration. Dorin Marcu, Vu Le, and Cristina Boicu contributed significantly to this research.

TABLE OF CONTENTS

1. RESEARCH OBJECTIVE AND APPROACH

This project has investigated an agent learning theory for mixed-initiative knowledge acquisition. The objective is to enable rapid development and easy maintenance of knowledge-based problem solving and decision-making assistants in uncertain and dynamic environments, cognitive assistants that can capture, use, preserve, and transfer to other users the military expertise which currently takes years to establish, is lost when military experts separate from service, and is costly to replace.

Such a cognitive assistant that helps with expert problem solving resembles an expert system. As it is well-known, the process of developing an expert system is long, difficult and error-prone (Buchanan and Wilkins, 1993; Tecuci and Kodratoff, 1995). It involves a knowledge engineer who attempts to understand the reasoning of an expert, identify general reasoning patterns, verify them with the expert and represent them into the knowledge base of the expert system. Moreover, any necessary changes in the knowledge base have to be again performed by the knowledge engineer, and this knowledge base maintenance process is even more difficult and time-consuming than the initial development of the knowledge base. The result is a rigid system that cannot keep pace with the advancements in its area of expertise and rapidly loses its utility.

The type of cognitive assistant investigated in this project is qualitatively different from an expert system. Instead of being programmed by a knowledge engineer, the agent learns its expertise directly from an expert who can teach it in a way that is similar to how the expert would teach a student or a new collaborator, through problem solving examples and explanations. As a result of learning from the expert and from its own experience, the agent acquires complex reasoning rules from the expert, continuously extends its representation language, and continuously updates the previously learned rules to the extended representation language, with the goal of better capturing and representing the expert's knowledge, and becoming a more useful assistant. However, when such an agent is first used by an expert, it does not engage in this interaction with a blank mental tablet because it already has a stock of established knowledge about evidence, its properties, uses, and discovery (Schum, 1987; Schum 2001a; Schum et al., 2009a; Tecuci et al., 2009). Thus the agent does learn about specific problems from experts, but it also combines this knowledge with what it already knows about various elements of evidential reasoning. Conventional expert systems can be no better than the expertise of the persons whose heuristics are captured; this represents a 'ceiling' on their suitability. But this ceiling is actually the 'floor' for our agent, since it incorporates the complex knowledge of the evidential reasoning tasks experts face in addition to the substantive expertise captured from the experts.

Figure 1 summarizes our research approach. The developed knowledge representation, learning and reasoning methods have been implemented into a general agent building tool called the Disciple 2009 Learning Agent Shell. As shown in the upper right side of Figure 1, this agent shell can (and has been used to) acquire general textbook knowledge from the Science of Evidence (Boicu et al., 2008b; Schum et al., 2009a,b). This knowledge is invaluable in any domain which involves evidence-based reasoning.

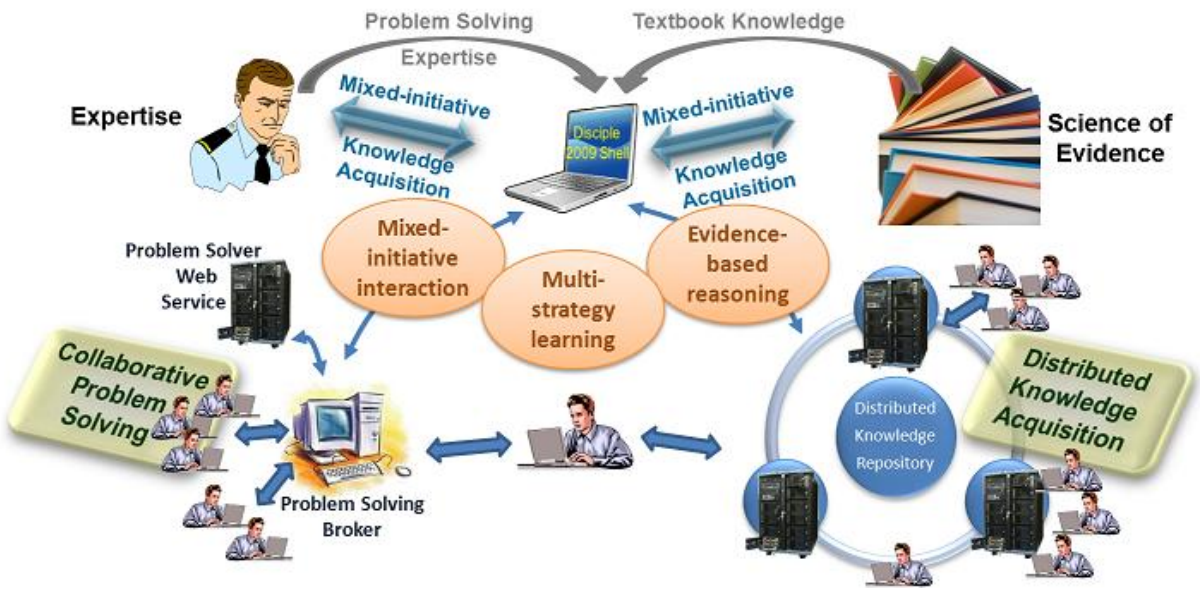

Problem Solving Expertise
Textbook Knowledge
Expertise
Mixed-initiative Knowledge Acquisition
Disciple 2009 Shell
Mixed-initiative Knowledge Acquisition
Science of Evidence
Problem Solver Web Service
Mixed-initiative interaction
Multi-strategy learning
Evidence-based reasoning
Collaborative Problem Solving
Problem Solving Broker
Distributed Knowledge Repository
Distributed Knowledge Acquisition

Figure 1. Overview of the research approach.

The agent shell can now be rapidly trained by a subject matter expert to solve problems in a specific domain (e.g., intelligence analysis), becoming a specific cognitive assistant, as indicated in the upper left side of Figure 1 (Tecuci et al., 2007c, 2008a, 2008b). The knowledge base of this Disciple cognitive assistant can be even more rapidly developed through distributed knowledge acquisition, as indicated in the lower right side of Figure 1 and discussed in (Tecuci and Boicu, 2008a). This process reduces most of the software engineering activities of developing a new knowledge-based cognitive assistant to more efficient knowledge engineering activities of extending the knowledge base of the Disciple learning agent shell.

An end-user (shown in the middle-bottom of Figure 1) can now use the cognitive assistant through mixed-initiative interaction (Tecuci et al, 2007a,b; Marcu 2009), where they interleave their contributions to a joint reasoning process in a natural way, based on their relative knowledge, rather than by following a fixed interaction pattern. The agent will use its knowledge from the Science of Evidence to support its user in solving complex evidence-based reasoning problems in partially known and dynamic environments (Boicu et al., 2008b; Schum et al., 2009b; Tecuci et al., 2009). Its divide-and – conquer approach to problem solving facilitates collaborative problem solving, as indicated in the lower-left part of Figure 1.

During the reasoning process, the agent will show very clearly what evidence was used and how, what is not known, and what assumptions have been made. It can also educate its user on all the aspects of the problem solving process (Le 2008; Le et al, 2008a). Through multistrategy learning, the agent will also be able to personalize itself to specific users, significantly facilitating its acceptance by them.

The next sections summarize each of these research results.

2. PROBLEM CONTEXT: EVIDENCE-BASED REASONING

Problem solving and decision-making in uncertain and dynamic environments involve evidence-based reasoning because, generally, one has to reason with evidence about facts rather than the facts themselves. Therefore we have researched agent learning for mixed-initiative knowledge acquisition in the context of evidence-based reasoning problems, particularly intelligence analysis, which is so important for the national defense.

2.1 Discovery of Hypotheses, Evidence and Arguments: Reasoning and Learning

We view evidence-based reasoning as a process of ceaseless discovery in a non-stationary world, process involving evidence in search of hypotheses (through abductive reasoning), hypotheses in search of evidence (through deductive reasoning), and evidential tests of hypotheses (through inductive reasoning), all going on at the same time (Schum et al., 2009a; Tecuci et al., 2009). As we know, *abduction* shows that something is *possibly* true, *deduction* shows that something is *necessarily* true, and *induction* shows that something is *probably* true.

Figure 2 summarizes our general approach to evidence-based reasoning and learning. The discovery of a surprising item of evidence (E^*_i) leads someone to abductively leap to a hypothesis of interest H_k. In this case we have *evidence in search of hypotheses* where the person experiences a flash of insight and generates this hypothesis. Asked to justify it, he or she provides the *abductive reasoning* chain from the left hand side of Figure 2.

The diagram in the middle of Figure 2 illustrates the deductive processes involved when we have *hypotheses in search of evidence*. Once the new hypothesis H_k has been generated, the person has to assess it, through *deductive reasoning*. The reasoning might start as follows. If H_k were true, there are sub-hypotheses, listed as H_d and H_e, that would be necessary and sufficient to make H_k true. If H_d were true then one would need to observe E_p. Similarly, each of the sub-hypotheses allows that person to deduce potential items of evidence (shown as shaded circles) that bear upon them. So here we have hypotheses in search of evidence that guides the process of collecting new evidence.

Now, some of the newly discovered items of evidence may trigger new hypotheses (or the refinement of the current hypothesis). So, as indicated at the bottom left of Figure 2, the processes of evidence in search of hypotheses and hypotheses in search of evidence take place at the same time, and in response to one another.

The combination of evidence in search of hypotheses and hypotheses in search of evidence results in hypotheses which have to be tested, through *inductive reasoning*, based on the discovered items of evidence, as illustrated in the right-hand side of Figure 2 with hypothesis H_k. The result of the testing process is the likelihood of the considered hypothesis.

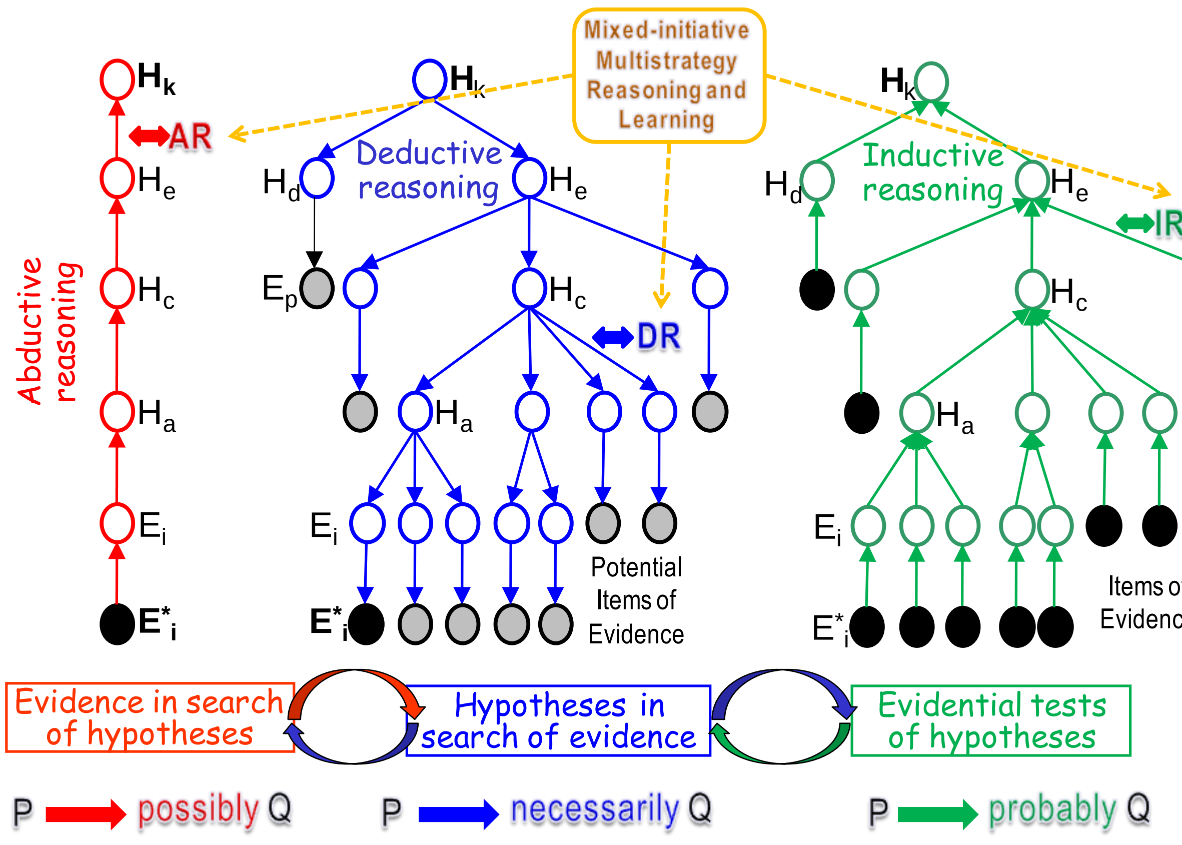

Mixed-initiative Multistrategy Reasoning and Learning
Abductive reasoning
AR
H_k
H_e
H_c
H_a
E_i
E^*_i
Deductive reasoning
DR
H_d
E_p
Potential Items of Evidence
Inductive reasoning
IR
Items of Evidence
Evidence in search of hypotheses
Hypotheses in search of evidence
Evidential tests of hypotheses
P possibly Q
P necessarily Q
P probably Q

Figure 2. Evidence-based reasoning and learning.

Our goal is to enable an agent to learn to perform this kind of reasoning, directly from a human expert, so that it can effectively assist the expert and other users with future evidence-based reasoning tasks. In particular, from each reasoning step (be it abductive, deductive, or inductive), the agent should attempt to learn a general (abductive, deductive, or inductive) rule, through a mixed-initiative multi-strategy learning process.

We have developed methods for performing these kinds of deductive and inductive reasoning, as well as for learning deductive rules and inductive rules. Abductive reasoning and learning cannot be fully automated, requiring human participation. However, as we will discuss in Section 11, a future research direction is the development of methods for assisting a user in performing abduction, as well as methods for learning some types of abductive rules (Eco 1983; Thagard, 1993; Schum 2001b).

2.2 Wigmorean Networks for Hypothesis Analysis: Logic and Probabilistic Reasoning

Figure 3 illustrates the integrated deductive and inductive reasoning performed by a Disciple 2009 cognitive assistant and a user in order to assess the likelihood of a hypothesis. This is a Wigmorean probabilistic inference network that combines the deductive reasoning tree and the inductive reasoning tree from Figure 2. This network has a well-defined structure, which has a grounding in the problem reduction representations developed in Artificial Intelligence (Nilsson, 1971; Tecuci, 1988; Tecuci, 1998), and in the argument construction methods provided by the noted jurist John H. Wigmore (1937), the philosopher of science Stephen Toulmin (1963), and the evidence professor David Schum (1987, 2001a).

This approach uses expert knowledge and evidence to successively reduce a complex hypothesis analysis problem to simpler and simpler problems, to find the solutions of the simplest problems, and to compose these solutions, from bottom-up, to obtain the solution of the initial problem. The solutions of these problems are probabilistic because the evidence is always *incomplete* (no matter how much we have), usually *inconclusive* (it is consistent with the truth of more than one possible hypothesis), frequently *ambiguous* (we cannot always determine exactly what the evidence is telling us), commonly *dissonant* (some of it favors one possible hypothesis but other evidence favors other hypotheses), and with any gradation of *believability* shy of perfection (Schum, 1987; Schum, 2001a).

In our research we have considered *symbolic probabilities*, as indicated in Figure 4, with names similar to those from the US National Intelligence Council's standard estimative language. For example, as shown in Figure 4, indicating that a hypothesis is "likely" is equivalent to saying that its probability of being true is between 0.55 and 0.75. Of course, the actual symbolic probabilities and the associated intervals from Figure 4 are just examples. A user may decide to use other names for symbolic probabilities, as well as other associated intervals, as discussed by Kent (1994) and Weiss (2008).

The Wigmorean network from Figure 3 shows how evidence is linked to hypotheses through arguments that establish the *relevance*, *believability* and *inferential force or weight* of evidence (Schum et al., 2009a, 2009b). First, the assessment of hypothesis H_k (problem [P1]) is reduced to the assessment of two simpler hypotheses, H_d and H_e (problems [P2] and [P2], respectively). Each of these hypotheses is assessed by considering both *favoring evidence* and *disfavoring evidence* (e.g., problems [P5] and [P6]). Let us assume that there are two items of favoring evidence for H_d: E_1 and E_2. For each of them (e.g., E_1) the agent assesses the extent to which it favors the hypothesis H_d (i.e., [P6]). This requires assessing both the *relevance* of E_1 to H_d (problem [P8]) and the *believability* of E_1 (problem [P9]).

The *relevance* answers the question: *So what? How does this datum or item of information, whatever it is, bear on what a user is trying to prove or disprove?*

The *believability* answers the question: *Can we believe what this item of information is telling us?*

Let us assume that the agent has obtained the following solutions for the problems [P8] and [P9]:

If we believe E_1 then H_d is almost certain. *[S8]*
It is likely that E_1 is true. *[S9]*

By compositing the solutions [S8] and [S9] (e.g., through a "min" function) the agent assesses the *inferential force or weight* of E_1 on H_d:

Based on E_1 it is likely that H_{11} is true. *[S6]*

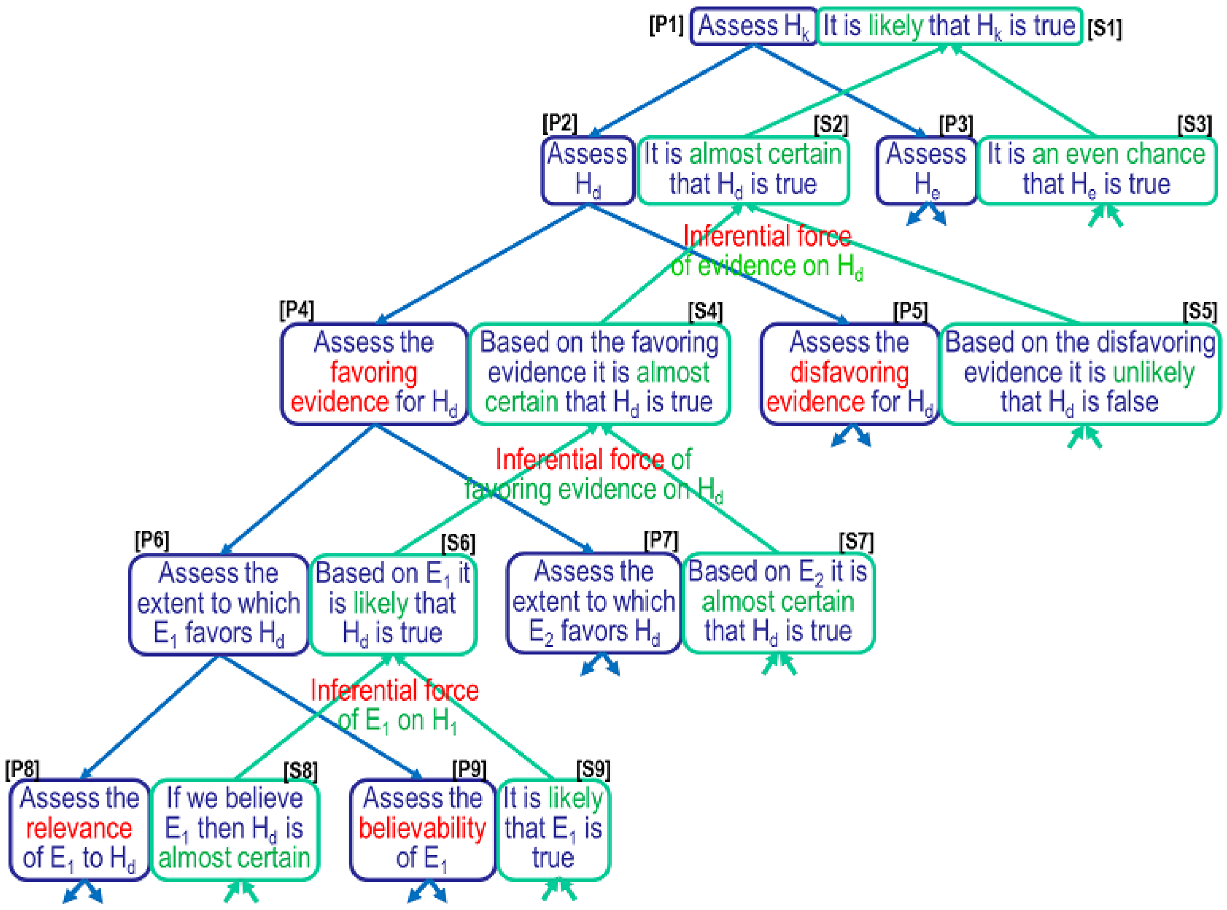

[P1] Assess H_k
It is likely that H_k is true [S1]
[P2] Assess H_d
[S2] It is almost certain that H_d is true
[P3] Assess H_e
[S3] It is an even chance that H_e is true
Inferential force of evidence on H_d
[P4] Assess the favoring evidence for H_d
[S4] Based on the favoring evidence it is almost certain that H_d is true
[P5] Assess the disfavoring evidence for H_d
[S5] Based on the disfavoring evidence it is unlikely that H_d is false
Inferential force of favoring evidence on H_d
[P6] Assess the extent to which E_1 favors H_d
[S6] Based on E_1 it is likely that H_d is true
[P7] Assess the extent to which E_2 favors H_d
[S7] Based on E_2 it is almost certain that H_d is true
Inferential force of E_1 on H_1
[P8] Assess the relevance of E_1 to H_d
[S8] If we believe E_1 then H_d is almost certain
[P9] Assess the believability of E_1
[S9] It is likely that E_1 is true

Figure 3. Hypothesis analysis through logic and probabilistic reasoning.

As illustrated above, the *inferential force or weight* answers the question: *How strong is this item or body of relevant evidence in favoring or disfavoring various alternative hypotheses or possible conclusions being entertained?*

Similarly the agent assesses the inferential force or weight of E_2 on H_d:

Based on E_2 it is almost certain that H_{11} is true. *[S7]*

By composing the solutions [S6] and [S7] (e.g., through a "max" function) the agent assesses the inferential force/weight of the favoring evidence (i.e., E_1 and E_2) on H_d:

Based on the favoring evidence it is almost certain that H_d is true. *[S4]*

Through a similar process the agent assesses the disfavoring evidence for H_d:

Based on the disfavoring evidence it is unlikely that H_{11} is false. *[S5]*

Because there is very strong evidence favoring H_d and there is weak evidence disfavoring H_d, the agent concludes:

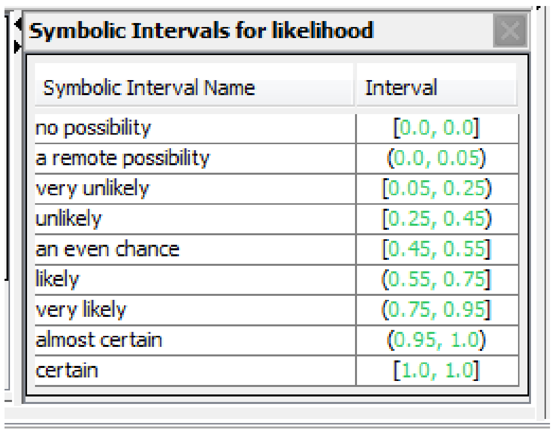
It is almost certain that H_d is true. [S2]

The sub-hypothesis H_e is assessed through a similar process:

It is an even chance that H_e is true. [S3]

The solutions of H_d and H_e are composed (e.g., through "average") into the evidence-based assessment of H_k:

It is likely that H_k is true. [S1]



Symbolic Intervals for likelihood

Symbolic Interval Name	Interval
no possibility	[0.0, 0.0]
a remote possibility	(0.0, 0.05)
very unlikely	[0.05, 0.25)
unlikely	[0.25, 0.45)
an even chance	[0.45, 0.55]
likely	(0.55, 0.75]
very likely	(0.75, 0.95]
almost certain	(0.95, 1.0)
certain	[1.0, 1.0]

Figure 4. Sample symbolic probabilities and their corresponding probability intervals.

This reasoning approach of the Disciple agent, illustrated in Figure 3, represents a *natural integration of logic and probabilistic reasoning.* This is in itself a significant accomplishment because most of the work on the development of cognitive systems has been done either in a logical framework [e.g. CYC (Lenat, 1995), Protégé (Musen et al., 2003), TAINS/TRIPS (Ferguson and Allen, 2007), MAPGEN (Bresina and Morris, 2007), PExA (Myers et al., 2007), Disciple (Tecuci, 1998)], or a probabilistic one [e.g., (AUTOCLASS (Cheesman et al., 1988), MSBNx (Kadie et al., 2001), Bayesian KB (Santos and Dinh, 2004), Analytica (Arnold et al., 2007), JCAT (Lemmer, 2007)], with not much interaction between the two research communities. However, robust, flexible, and scalable problem-solving and decision-support systems for complex military and civilian applications require both logical and probabilistic representations, with corresponding learning and reasoning capabilities. Indeed, while logical and understandable solutions are needed, they have to be developed in the context of uncertain knowledge about a dynamic environment, which necessarily requires probabilistic reasoning.

Important recent work on the integration of the logical and the probabilistic paradigms has been reported in (Shirazi and Amir, 2007; Domingos and Richardson, 2006; Milch, et al., 2005; Nodelman, et al., 2005; Sutton et al., 2007). However, most of these studies concern conventional or Bayesian probabilistic models and assume that probabilities will always be supported by relative frequencies and relevant statistics. In other words, their assumption is that the events of concern are repeatable or replicable. They do not consider the properties, uses, and discovery of evidence about unique, singular, or one-of-a-kind events. Such evidence occurs frequently in very many military and civilian applications, including most intelligence analysis and the intelligence preparation of the battlefield.

3. LEARNING-ORIENTED KNOWLEDGE REPRESENTATION

We have developed a knowledge representation centered on the notion of concept that facilitates the basic operations involved in learning, such as generalizing concepts, specializing concepts, and comparing the generality of concepts. This representation is described in detail in (Tecuci and Boicu 2008c).

3.1 Concept Representation

Basic concepts representing objects and features are explicitly represented in an ontology (Tecuci and Boicu, 2008b). More complex concepts are expressed using first order logical expressions involving basic concepts. The basic representation unit (BRU) for a more complex concept has the form of a sequence ($?O_1$, $?O_2$,…, $?O_n$), where each $?O_i$ has the following structure, called ***clause***:

$?O_i$	instance of	concept_i
	feature_{i1}	$?O_{i1}$
	. . .	
	feature_{in}	$?O_{im}$

Concept_i is either an object concept from the object ontology, or a numeric interval, or a set of numbers, or a set of strings, or an ordered set of intervals. Feature $_{i1}$ … feature_{in} are features from the ontology. $?O_{i1}$ … $?O_{im}$ are distinct variables from the sequence ($?O_1$, $?O_2$, … , $?O_n$).

A concept may be a conjunctive expression of the following form, meaning that any instance of the concept satisfies BRU and does not satisfy BRU_1 and … and does not satisfy BRU_p:

BRU
Except-When BRU_1
. . .
Except-When BRU_p

Such expressions are learned from positive and negative examples, and can be easily generalized (to cover new positive examples) or specialized (to uncover negative examples) by using various generalization and specialization rules. The generalization rules include: turning constants into variables, turning occurrences of a variable into different variables, climbing the generalization hierarchies, dropping conditions, extending intervals, extending ordered sets of intervals, extending discrete sets, using feature definitions, and using inference rules. The reverse of these generalization rules are specialization rules.

Partially learned concepts are represented as plausible version spaces, as illustrated in Figure 5. The concept in Figure 5 is characterized by a main plausible version space condition and may also have one or several except-when plausible version space conditions. Each plausible version space is defined by a plausible lower bound and a plausible upper bound. The plausible lower bound of the main plausible version space condition is a minimal generalization of the positive examples of the concept that covers as few negative examples as possible. Similarly, the plausible upper bound is a maximal generalization. These generalizations are obtained by applying the generalization and specialization rules mentioned above. These generalizations are called plausible because they are based on the incomplete ontology of the agent which constitutes the agent's generalization hierarchy (Tecuci and Boicu, 2008b). Each except-when plausible version space condition characterizes a set of negative examples of the concept. Therefore, the concept represented in Figure 5 is the set of examples that are covered by the main condition without being covered by the except-when condition. As indicated in Figure 5, this representation associates levels of likelihood to its potential instances. For example, an instance covered by the lower bound of the main plausible version space condition (showed as dark green in Figure 5) is most likely a positive example. Similarly, an instance covered by the lower bound of an except-when condition (showed as dark red), is most likely a negative example. The representation language also allows explicit representation of the known positive and negative exceptions of the concept.

All the elements of the knowledge base of an agent are represented as concepts, as discussed in the following sections.

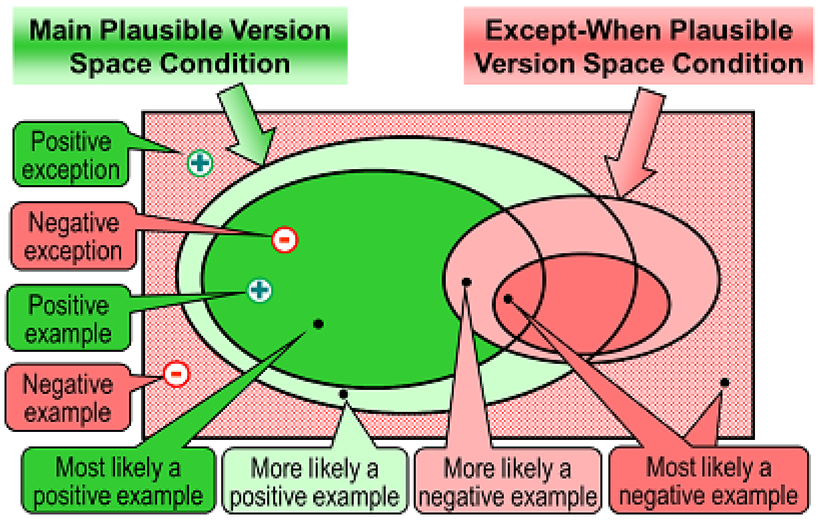

Main Plausible Version Space Condition
Except-When Plausible Version Space Condition
Positive exception
Negative exception
Positive example
Negative example
Most likely a positive example
More likely a positive example
More likely a negative example
Most likely a negative example

Figure 5. Partially learned concept with exceptions.

3.2 Knowledge Base

The knowledge base of an agent consists of an object ontology and various types of reasoning rules.

3.2.1 Ontology

A detailed description of the ontological representation, the ontology development tools, and the knowledge engineering guidelines for building ontologies is provided in (Boicu et al., 2008b; Tecuci and Boicu 2008c).

The ontology is a hierarchical representation of the objects from the application domain of an agent. It includes both descriptions of the different types of objects (called concepts, such as tangible evidence or authoritative record), and descriptions of individual objects (called instances, such as Evidence-1), together with the properties of each object and the relationships between objects. A fragment of an ontology of evidence is shown in the right hand side of Figure 6.

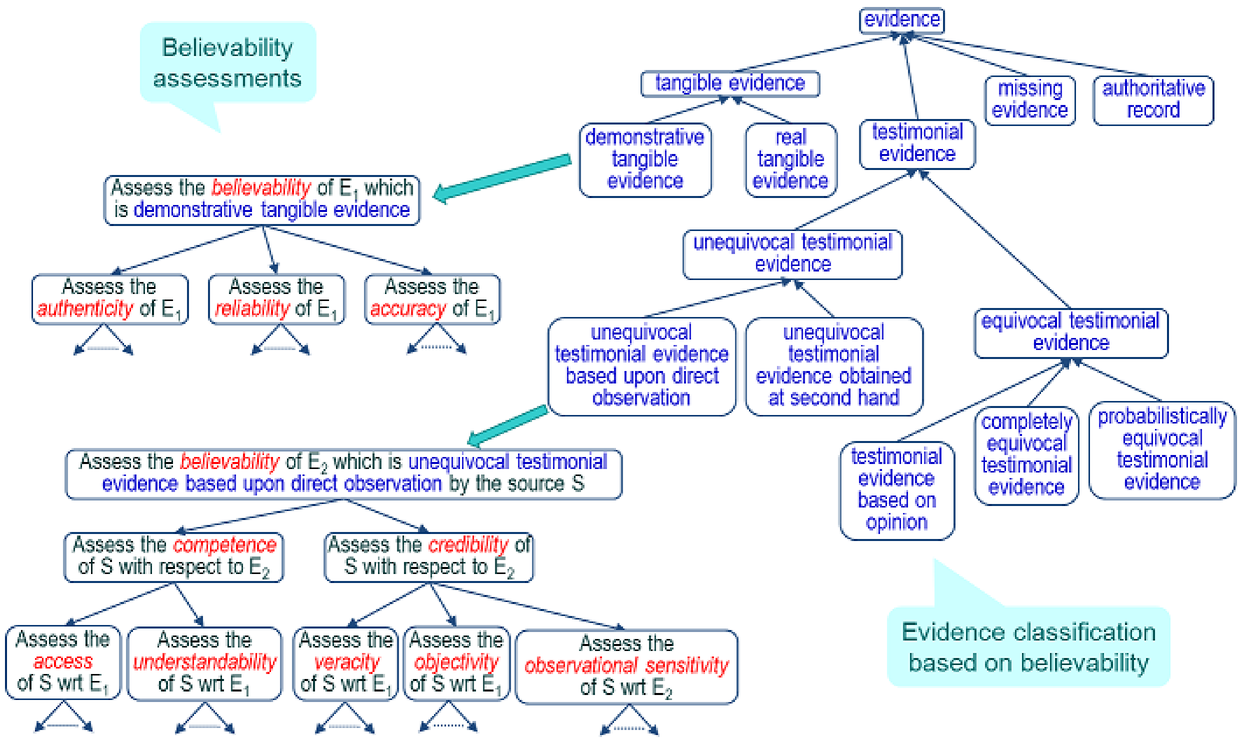

Believability assessments
evidence
tangible evidence
missing evidence
authoritative record
demonstrative tangible evidence
real tangible evidence
testimonial evidence
Assess the believability of E1 which is demonstrative tangible evidence
unequivocal testimonial evidence
Assess the authenticity of E1
Assess the reliability of E1
Assess the accuracy of E1
unequivocal testimonial evidence based upon direct observation
unequivocal testimonial evidence obtained at second hand
equivocal testimonial evidence
completely equivocal testimonial evidence
probabilistically equivocal testimonial evidence
testimonial evidence based on opinion
Assess the believability of E2 which is unequivocal testimonial evidence based upon direct observation by the source S
Assess the competence of S with respect to E2
Assess the credibility of S with respect to E2
Assess the access of S wrt E1
Assess the understandability of S wrt E1
Assess the veracity of S wrt E1
Assess the objectivity of S wrt E1
Assess the observational sensitivity of S wrt E2
Evidence classification based on believability

Figure 6. Evidence classification and its credentials.

The ontology plays a crucial role for a cognitive assistant, being at the basis of knowledge representation, user-agent communication, problem solving, knowledge acquisition and learning. First, the ontology provides the basic representational constituents for all the elements of the knowledge base, such as the problems, the problem reduction rules, and the solution synthesis rules. It also allows the representation of partially learned knowledge, based on the plausible version space concept.

Second, the agent's ontology enables it to communicate with the user and with other agents by declaring the terms that the agent understands. Consequently, the ontology enables knowledge sharing and reuse among agents that share a common vocabulary which they understand.

Third, the problem solving methods of the agent are applied by matching them against the current state of the agent's world which is represented in the ontology. The use of partially learned knowledge (with plausible version spaces) in reasoning allows solving of problems with different degrees of confidence.

And fourth, the ontology represents the generalization hierarchy for learning, where specific problem and problem solving steps are generalized into rules by replacing instances with concepts expressed with the objects and features from the ontology.

The ontology of the agent is considered to be incomplete and is continuously extended during the agent's problem solving and learning. This means that new types of objects may be added to the ontology, as well as new types of features, individual objects and individual features. A feature is defined by its domain (the type of objects that can have that feature) and its range (the set of possible values). Both the domains and the ranges are concepts that can be partially learned and are continuously evolving.

3.2.2 Reasoning Rules

Problems, Problem Reductions, Solutions Synthesis

As discussed in Section 2.2, the agents considered in this research employ a general divide-and-conquer approach to problem solving which we refer to as problem reduction and solution synthesis. As Alvin Toffler (1984) pointed out: "One of the most highly developed skills in contemporary Western civilization is dissection; the split-up of problems into their smallest possible components. We are good at it. So good, we often forget to put the pieces back together again." Indeed, this paradigm was successfully applied to a wide range of domains (e.g., planning, analysis, design, programming, medical diagnosis), but most of the approaches emphasize the reduction phase, where complex problems are successively reduced to simpler ones. Most often the synthesis phase is reduced to either selecting the solution of a simpler problem as the solution of the initial problem, as in medical diagnosis (Ahn et al., 2006), or to collecting all the elementary solutions, as in hierarchical task-network planning (Ghallab et al., 2004). Our approach is general both with respect to the reduction phase and with respect to the synthesis phase.

As shown in Figure 7, Problem 1 can be solved through different strategies, each reducing it to a different set of subproblems. For instance, if Question 1 has as answer Answer 1, then to solve Problem 1 one has to solve Problem 2 and Problem 3. Moreover, if Question 2 has as answer Answer 2, then there is an additional way of solving Problem 1, by solving Problem 4 and Problem 5. Once the solutions of Problem 2 and Problem 3 have been found (i.e. Solution 2 and Solution 3), they can be combined into Solution A1, corresponding to the first problem solving strategy. Similarly, one obtains Solution A2, corresponding to the second problem solving strategy. Then, in order to find the solution of Problem 1, one has to combine Solution A1 and Solution A2. Thus, in general, to obtain the solution of a problem (e.g. Problem 1) one not only needs to obtain the solution corresponding to each reduction strategy (by using a reduction-level synthesis), but also to combine all these solutions into the solution of the problem (by using a problem-level synthesis). Moreover, the agent would need to be able to provide a solution to Problem 1 when only some of the attempted reduction strategies provide solutions, which requires additional types of problem-level synthesis.

As this example illustrates, the knowledge base of the agent has to include problem reduction rules, reduction-level synthesis rules, and problem-level synthesis rules which, when applied, generate reasoning trees like the ones in Figure 3 and Figure 7. The representation of these different types of rules is discussed in the next section.

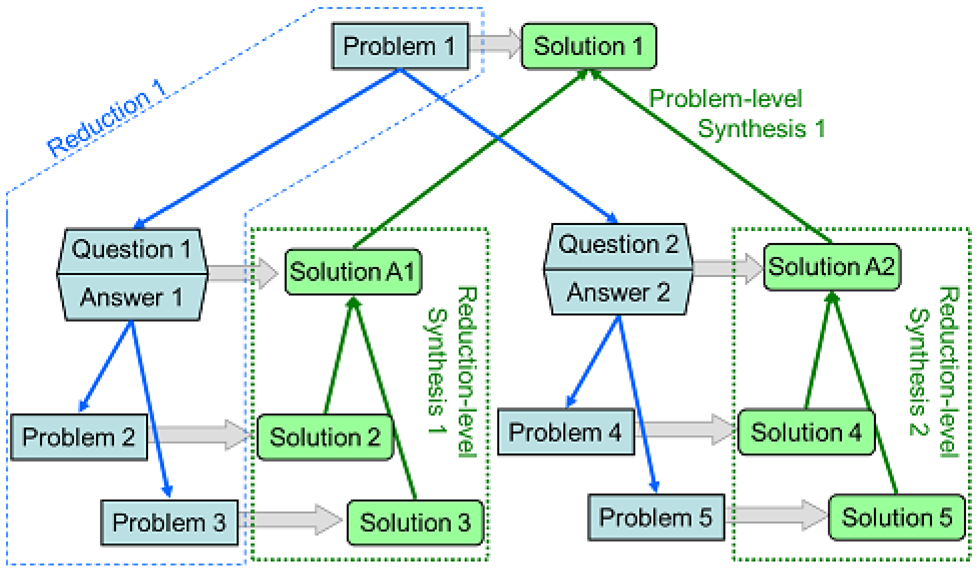


Figure 7. Problem reduction and solution synthesis.

Concept-based Representation of Different Types of Rules

The researched agents need to reason formally with the informal natural language used by the experts and the end-users. To accomplish this, we have defined a unified structure for all the agent's reasoning rules which includes of the following elements:

- An informal structure and associated parameters which preserves the natural language pattern of an expression such as a problem, a problem reduction step, or a solution synthesis step. This structure is generated by simply replacing each instance or constant from the natural language expression with a variable which becomes a structure parameter.

- A (partially learned) concept which represents the set of all possible combinations of parameter values for which the corresponding natural language structure is a correct expression (e.g., a problem that makes sense, or a correct problem reduction step).

- The examples and the explanations from which the rule (or concept) has been learned. They are kept in two forms which correspond to the organization of the knowledge base into a domain part (containing the rules, the concepts and the instances that are common to all the possible scenarios) and several scenario parts (each scenario containing only the instances that exist in that scenario). One form of an example (explanation) is a minimal generalization that does not contain any instance. This is kept in the domain part of the knowledge base. The other form is the actual example (explanation) which contains instances and is kept in the corresponding scenario part. This representation facilitates the efficient adaptation of the learned knowledge to changes in the agent's ontology, as discussed in the next section.

Learning and Problem Solving based on Confidence Levels

The left hand side of Figure 8 shows the structure of the reduction rule that would be learned from Reduction 1 in the left hand side of Figure 7. As illustrated in Figure 8, this rule has an applicability condition which is the concept covering all the instances for which the reduction is correct. A partially learned rule has a partially learned applicability condition (e.g., PVS Condition and Except-When PVS Condition). This partially learned rule generates reductions with different levels of plausibility.

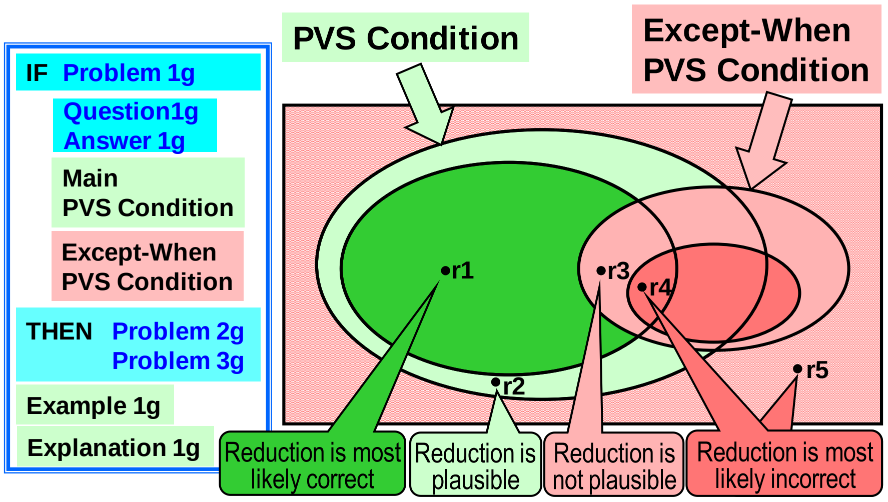


Figure 8. Plausible reasoning based on a partially learned rule.

The synthesis rules and the problems themselves have similar structures with applicability conditions that can be partially learned. This general structure is also used for other knowledge elements, such as partially learned features that are characterized by partially learned concepts representing their domains and ranges. This is important because the system can use the same learning methods for all the knowledge elements with this general structure. It can also use the same plausible reasoning methods based on various levels of confidence.

The way a partially learned rule is used depends on the main goal of the agent. If the main goal is to support its user in problem solving, then the agent will generate the solutions that are more likely to be correct. For example, a reduction covered by the plausible lower bound of the PVS condition and not covered by any of the Except-When PVS conditions (e.g. reduction r1 in Figure 8) will be preferable to a reduction covered by the plausible upper bound of the PVS condition and not covered by any of the Except-When PVS conditions (e.g., reduction 2), because it is more likely to be correct.

However, if the main goal of the agent is to improve its reasoning rules, then reduction r2 is more useful to be generated then reduction r1. Indeed, no matter how the user characterizes r2 (either as correct or as incorrect) the agent will be able to use it to refine the rule, either by generalizing the plausible lower bound of the PVS condition to cover r2 (if r2 is a correct reduction), or to specialize the plausible upper bound of the PVS condition to uncover r2 (if r2 is an incorrect reduction), or the learn an additional Except-When PVS condition based on r2 (again, if r2 is an incorrect reduction).

3.3 Generic Knowledge from the Science of Evidence

The emerging Science of Evidence, which is based upon 700 years of experience in the Anglo-American system of law for testing the competence and credibility of witnesses, provides significant knowledge for evidence-based reasoning. This general knowledge is not only very valuable for any agent performing evidence-based reasoning but it can be represented once in the knowledge base of an agent shell and used by any agent developed with that shell, without any additional knowledge acquisition effort.

We have collaborated with David Schum, one of the founding fathers of the Science of Evidence (Schum, 1987; Schum 2001a; Anderson et al., 2005), to acquire generic knowledge from the Science of Evidence and represent it into the knowledge base of the Disciple 2009 Learning Agent Shell.

Our paper (Schum et al., 2009b) presents a foundation for an evidence categorization scheme that tells us what kinds and combinations of evidence we have in any intelligence analysis regardless of the substance or content of the evidence and the objectives of the analysis. It presents three substance-blind classifications of evidence, one based on believability, one on relevance, and one on inferential force, which support the development of Wigmorean networks for hypotheses analysis (see Figure 3).

Part of the classification based on the believability of evidence is shown in the right hand side of Figure 6. The importance of this classification is that for each type of evidence identified (e.g., demonstrative tangible evidence, or unequivocal testimonial evidence based upon direct observation) there are specific credentials for assessing the believability or credibility of evidence (e.g., authenticity, reliability and accuracy for demonstrative tangible evidence), as well as specific and well-defined procedures for performing these assessments, as shown in the left hand side of Figure 6.

We have also developed a mixed-initiative computational approach for analyzing evidence and its chain of custody, which is presented in detail in (Schum et al, 2009a).

4. AGENT REASONING, TEACHING AND LEARNING

Instead of being programmed by a knowledge engineer, the agent learns its expertise directly from an expert who can teach it in a way that is similar to how the expert would teach a new collaborator, through problem solving examples and explanations. Figure 9 shows the most important phases of this process which are further discussed in the following sections.

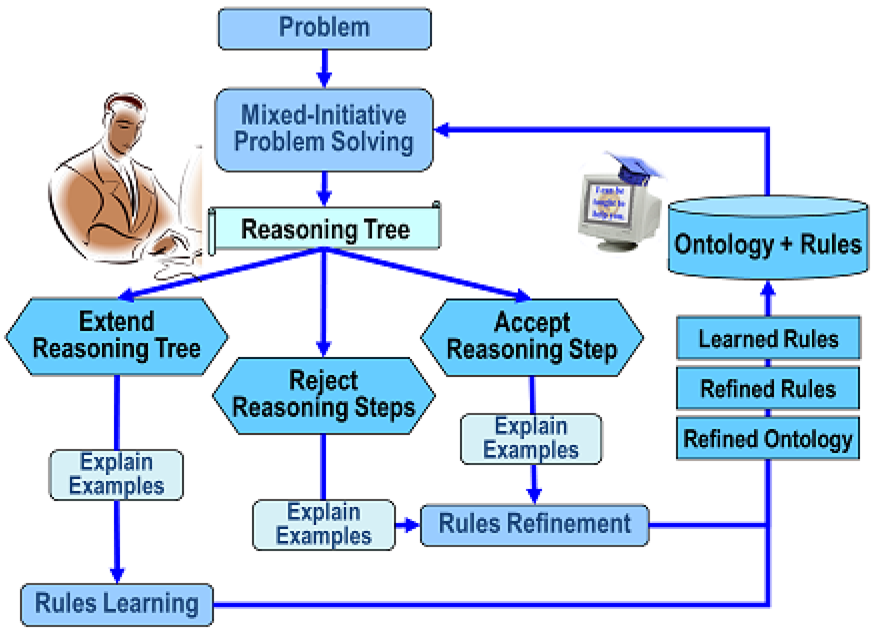


Figure 9. Control of modeling, learning and problem solving.

The expert formulates the problem to be solved and the agent uses its partially learned knowledge to solve it by generating a reasoning tree. The expert analyzes the generated tree and may take one of the following actions:

- If there is any sub-problem that the agent cannot solve by itself (i.e. a leaf in the generated reasoning tree), the expert interacts with the agent to develop a solution for that sub-problem and to explain it. As a result, the agent will learn new reasoning rules by generalizing the reasoning steps from the example solution sub-tree. The agent will also extend its ontology with the new instances and features encountered in the solution sub-tree or in the corresponding explanations.

- If there is any wrong reasoning step, the expert interacts with the agent to explain the mistake and the agent will specialize the rule which generated it, to no longer make such mistakes in the future. This case may also result in an extension of the ontology if the explanation of the mistake includes new instances or new features.

- The reasoning steps that are accepted by the expert as correct are used as positive examples of the corresponding reasoning rules which are generalized to cover them. In these cases the expert may also provide additional explanations which will specialize the rule and may extend the ontology.

While knowledge base development (and agent development in general) is known to be a difficult and time-consuming process, knowledge base refinement is even more difficult and time consuming. Notice, however, that in the case of the Disciple cognitive assistants there is no longer a distinction between knowledge base development and its refinement, both consisting of learning and refining the reasoning rules and the ontology.

5. MIXED-INITIATIVE REASONING

A main motivation of this research effort is the obvious complementariness between humans and computers with respect to complex problem solving and decision-making. Humans are slow, sloppy, forgetful, implicit, and subjective, but have common sense and intuition, and may find creative solutions in new situations. In contrast, computer systems are fast, rigorous, precise, explicit, and objective, but they lack common sense and the ability to deal with new situations. Moreover, in contrast to a computer system, a human has a very limited attention span and can analyze only a small number of alternatives at a time. Therefore, a main research objective was to create a mixed-initiative reasoning framework that synergistically integrates the complementary capabilities of humans and computer systems, taking advantage of their relative strengths to compensate for each-others weaknesses.

One element of this mixed-initiative reasoning framework is the developed divide-and-conquer (problem reduction and solution synthesis) approach to problem solving which is both natural for a human user and appropriate for an automated agent. As discussed in Section 6 this helps an expert to make explicit his or her reasoning process in solving a problem, and also allows the agent to help with this process.

A second element of this framework is represented by the developed mixed-initiative teaching and learning methods, where the expert helps the agent to learn (by providing representative examples and explanations) and the agent helps the expert to teach it by asking relevant questions whose answers guide it in learning and refining it knowledge (see Section 7).

A third element of the mixed-initiative reasoning framework is the actual control of the problem solving process where the human acts as the orchestrator of the reasoning process, guiding the high-level exploration, while the agent implements this guidance by taking into account the learned human's preferred problem solving strategies. To facilitate this process, we have developed a method to abstract the large reasoning tree generated by agent, allowing the user to easily understand and browse a much smaller abstract reasoning tree as a table of contents for the detailed tree (Le, 2008). As discussed in Section 8.2, we have also developed methods (implemented in an assumption assistant) that allow the user and the agent to solve problems based on incomplete and uncertain information, and in time-critical situations. In essence, the user can define solutions for specific problems as assumptions, when he or she does not have relevant evidence or time to decompose those problems (Marcu, 2009; Tecuci et al., 2007d).

A final element of this framework is the ability of the user to teach his/her agent how to interact, as discussed in Section 8.

6. MODELING EXPERT'S REASONING

Previous experiments have shown that making explicit their reasoning process is the most difficult activity during the teaching of the agent by an expert (Boicu et al., 2005). We have investigated two approaches to alleviate this difficulty. One is the development of guidelines for an expert to express his/her reasoning in the problem reduction / solution synthesis paradigm. The expert will express the problem to be solved in natural language and then will ask a question related to how to solve this problem (for example, what problem strategy to use or how to apply that strategy, once it was selected). Then the expert will provide the answer to that question which will guide him/her in formulating the reduction of the problem to simpler problems (see the reduction steps in Figure 7). It is as if the expert is thinking allowed while solving a problem. The questions and their answers force the expert to make explicit the reasons behind his/her solutions, and significantly facilitate agent's learning (see Section 7). Experience with the development of several Disciple cognitive assistants (see Section10) has shown that the experts find this form of expressing their reasoning very natural.

A second approach for alleviating the difficulty the experts face in making explicit their reasoning is the development of methods (implemented as a Modeling Assistant) that will guide them in this process. These methods and the developed Modeling Assistant are described in detail in (Marcu, 2009). The Modeling Assistant can make both specific and general suggestions on how to complete a current reduction step being defined by the expert. It uses analogy with previously learned rules to suggest a specific question for a problem, or a specific answer for a question, or specific sub-problems and solutions once the other elements of the reasoning step have been defined (Marcu, 2009; Boicu, 2002). Upon acceptance, a suggestion can be further refined by the expert. General suggestions are based on abstractions defined for the problems and reductions used to generate the reasoning tree (Le, 2008; Le et al., 2008a). The user is guided in selecting an abstraction of the current problem or reduction to be defined. Once an abstraction is selected, the system may suggest reasoning patterns that are covered by that abstraction.

The Modeling Assistant not only helps the experts in expressing their reasoning, but it also leads to better formalized knowledge bases by encouraging the reuse of the reasoning patterns as opposed to defining similar but slightly different patterns and associated rules.

7. RULE LEARNING AND REFINEMENT

7.1 Rule Learning

A significant accomplishment of this research is the generalization of the previously developed multistrategy apprenticeship learning methods. In particular, these generalized methods allow the learning of rules for different types of knowledge elements: problems, reduction steps, synthesis steps, domains and ranges of features (see Section 3.2.2). This is a type of integrated learning from examples, from explanations, and from experimentation, where the agent generalizes a natural language statement provided by the expert (e.g. a problem or a problem reduction step) into a general pattern by replacing all the instances and constants with variables and by learning a general concept that represents the set of instantiations of those variables for which the instantiated pattern makes sense (e.g., is a problem that makes sense or is a correct reduction of a problem to simpler problems).

Let us consider, for instance, the reasoning tree from Figure 7 formulated by the expert when teaching the agent. From each problem (e.g., Problem 1), the agent will learn a general rule of the form "IF <condition> THEN Problem 1g". The condition allows the instantiation of the variables from Problem1g such that the resulting problem makes sense. The condition of a partially learned rule will have the form illustrated in Figure 5.

Similarly, from each reasoning step, consisting of a problem, a question, its answer and one or several sub-problems, the agent learns a general reasoning rule which has the structure illustrated in Figure 8. The initial version space of the applicability condition is defined by two generalizations of the question-answer pair which represents the explanation of why the corresponding problem was reduced to the simpler problems: 1) the minimal generalization of the question-answer pair that does not contain any instance, and 2) the maximal generalization (Tecuci et al., 2007c).

7.2 Rule Refinement

The learned rules are immediately applied in problem solving, in situations where their plausible upper bound conditions are satisfied. These applications have then to be judged by the expert as being correct or incorrect, as illustrated in Figure 9. Moreover, in case of incorrect applications, the agent will interact with the expert to find the explanation of why the corresponding reasoning step is incorrect, and will use this explanation (if found) to refine the rule. There are many strategies to refine a rule to no longer cover a negative example, as described in (Boicu et al., 2007) and illustrated in Figure 10.

As one can notice in Figure 10, there are many regions in the condition space where the negative example can be located and for each of them there are several alternative ways of refining the rule, depending on the form of the failure explanation. Let us consider, for instance, that the negative example is covered by the plausible upper bound of the main condition but it is not covered by the plausible lower bound. If the failure explanation has the form "I_1 is C_1", where I_1 is an instance from the example and C_1 is a concept from the ontology, then the agent can specialize the plausible upper bound of the main condition to no longer cover the negative example. If, however, the failure explanation has a different form (e.g. "object1 relation object2") then the agent will add an except-when plausible version space condition to the rule. If no failure explanation is found, then the agent will keep the negative example as a negative exception.

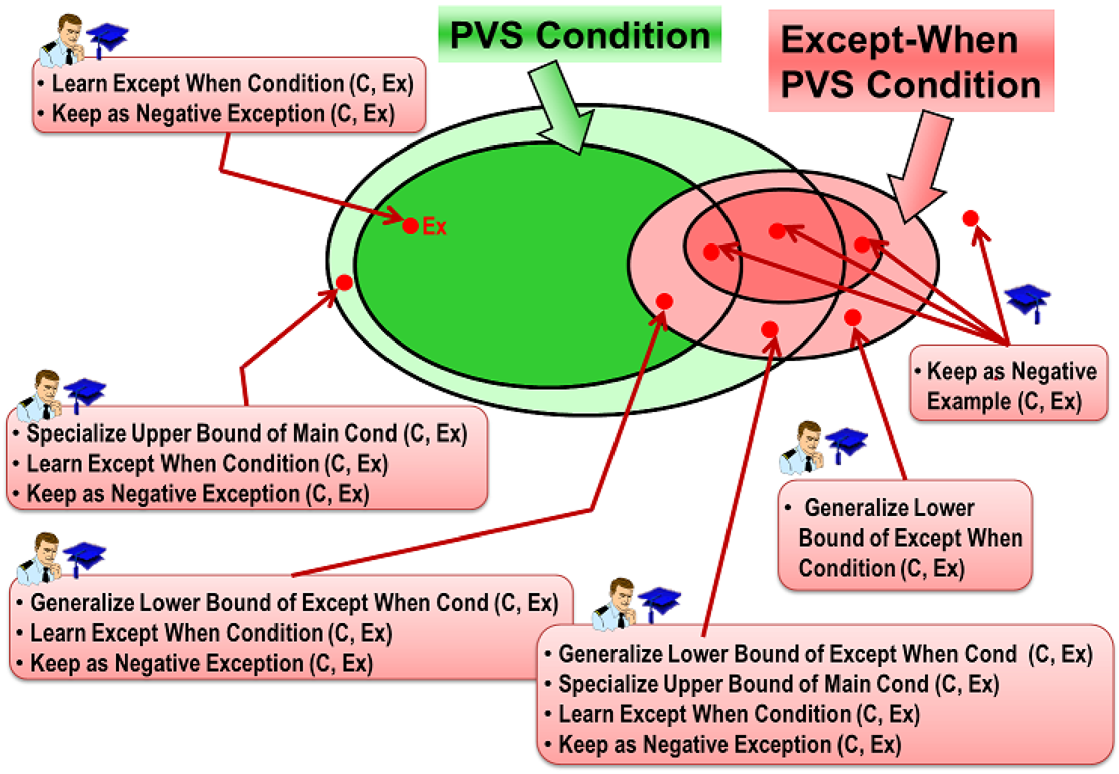

PVS Condition
Except-When PVS Condition
• Learn Except When Condition (C, Ex)
• Keep as Negative Exception (C, Ex)
Ex
• Specialize Upper Bound of Main Cond (C, Ex)
• Learn Except When Condition (C, Ex)
• Keep as Negative Exception (C, Ex)
• Generalize Lower Bound of Except When Cond (C, Ex)
• Learn Except When Condition (C, Ex)
• Keep as Negative Exception (C, Ex)
• Keep as Negative Example (C, Ex)
• Generalize Lower Bound of Except When Condition (C, Ex)
• Generalize Lower Bound of Except When Cond (C, Ex)
• Specialize Upper Bound of Main Cond (C, Ex)
• Learn Except When Condition (C, Ex)
• Keep as Negative Exception (C, Ex)

Figure 10. Rule refinement with negative examples.

7.3 Learning and Adaptation in a Dynamic Environment

As discussed in Section 3.2.2, the agent stores with a learned rule minimal generalizations of the examples and the explanations from which the rule was learned, generalizations that do not contain any instances or constants (see Example 1g and Explanation 1g in Figure 8). The generalized examples will also store their current support as the number of the known specific examples they covers in all the scenarios. If a generalized example has only positive examples (e.g., correct problem reductions) or only negative examples (e.g., incorrect problem reductions), then it is a positive general example and, respectively, a negative general example. If a generalized example has both positive and negative specific examples that it is considered irrelevant for relearning because nothing was stored to differentiate between the examples. This allows the agent to store valuable relearning information in an efficient way, without storing the scenarios containing the actual examples, which would require a very large amount of memory. With this information, the agent is able to continuously and efficiently update the previously learned rules in response to changes in the ontology, such as the introduction of new concepts and relationships. For example, a previously learned rule can be automatically relearned from its general examples and explanations, in the context of the new ontology, as will be described below. In principle, this is a time-consuming process because it needs to be performed each time the ontology is changed. However, we have developed a very efficient method, where version numbers are associated with the ontology and with the individual rules. When a change is made in the ontology, its version number is incremented. When a rule is used in problem solving and its version number is different from that of the current ontology, the rule is automatically relearned and its version number updated to that of the ontology. This simple scheme assures that rules updating takes place only when needed and only for the rules that are needed, rather than for each rule after each change of the ontology.

Rule relearning has three main phases. In the first phase the minimally generalized examples and explanations are validated in the context of the modified ontology. The specific examples corresponding to the available scenarios are also validated. The validation will check if the elements from examples and the explanations are still in the ontology. If not, they are either removed from the corresponding examples/explanations, or the entire examples/explanations containing them are removed. For the existing specific examples, the corresponding general examples are also checked to determine whether they are still minimal generalizations, updating them, if necessary. The support associated with the general examples is also updated.

The second phase of a rule's relearning re-computes its new plausible version space condition. The lower bound of the main plausible version space condition is re-computed as the minimal generalization of the positive general examples that does not cover the general negative examples. Similarly, the upper bound of the main plausible version space condition is re-computed as a maximal generalization. The new except when plausible version space conditions are computed in a similar way as minimal/maximal generalizations on the general negative examples that do not cover the general positive examples.

In the third phase, after the entire plausible version space condition of a rule was re-computed, the general examples as well as the available specific examples (i.e., those for which the corresponding scenarios are still available) are re-checked and positive and negative exceptions are determined. The positive exceptions are specific or general positive examples that are not covered by the updated upper bound of the main condition of the rule. The negative exceptions are specific or general negative examples that are covered by the lower bound of the main condition without being covered by any except-when condition. These exceptions are generally caused by the incompleteness of the ontology, and they will guide its future extensions.

8. LEARNING MIXED-INITIATIVE USER-AGENT INTERACTION MODELS

8.1 An Approach to Learning Interaction Models

Mixed-initiative interaction facilitates a collaboration between intelligent agents and their users that takes advantage of their complementary capabilities by supporting each of them in taking the initiative to perform the tasks for which they are most qualified, at the appropriate time. We have researched a learning-based approach to the development of mixed-initiative models of the interaction between an end-user and a cognitive assistant. This approach, described in more detail in (Marcu, 2009), is characterized by:

- An architecture for a cognitive assistant implemented as a collection of agents based on mixed-initiative interaction models.
- A step-by-step process for developing additional mixed-initiative interaction agents that can be integrated into the architecture of the cognitive assistant to extend its functionality.

At the basis of our approach is a task analysis methodology that results in the learning of executable interaction models by the mixed-initiative assistants. It includes general methods and guidelines for developing conceptual plans for human-agent interaction, translating these conceptual interaction plans into interaction models executable by a state-based interaction engine, the conceptualization of the interactions into an interaction ontology, and the adaptation and application of our learning methods to enable the learning of general interaction patterns.

This approach transforms the software engineering activity of programming interaction models into a knowledge engineering one. It was used in the development of two innovative mixed-initiative interaction agents, the Modeling Assistant discussed in Section 6 and the Assumption Assistant discussed in this section. Both these assistants are part of the multi-agent Disciple cognitive assistant.

8.2 Mixed-Initiative Assumption Assistant

The Assumption Assistant helps Disciple users in solving problems based on incomplete and uncertain information by allowing them to define solutions for unsolved problems as assumptions, to override agent-generated solutions, and to experiment with “What-If” scenarios.

Let us consider, for example, the actions that must be performed by the user while interacting with the Assumption Assistant to define assumptions for problems with no solution from a generated reasoning tree. A possible interaction between the user and the agent is shown in the left side of Figure 11. In essence, the interaction consists in selecting a problem for which to define a new assumption, invoking the new assumption editing component, determining the applicable solution patterns and defining the new assumption by instantiating the desired pattern, followed by saving the new assumption and continuing the same process for the other problems with no solution. The actions performed by the expert are shown with white background while those performed by the agent are shown with light blue background.

In the regular (non mixed-initiative) interaction, shown in the left hand side of Figure 11, the user has to perform almost all of the actions. However, if the agent understands the current reasoning context (in this case that of problem solving, as opposed to rule learning or modeling expert’s reasoning), it can take the initiative in performing more operations, as illustrated in the right side of Figure 11. Because the agent performs more actions, the user may concentrate on the more creative operations. This is expected to lead to a much more efficient interaction and a better user experience. In our approach, the user can teach the agent how he or she would like to interact with it, and the agent learns general interaction models from the user. This learning ability allows the agent to also adapt to the changing needs and preferences of the user.

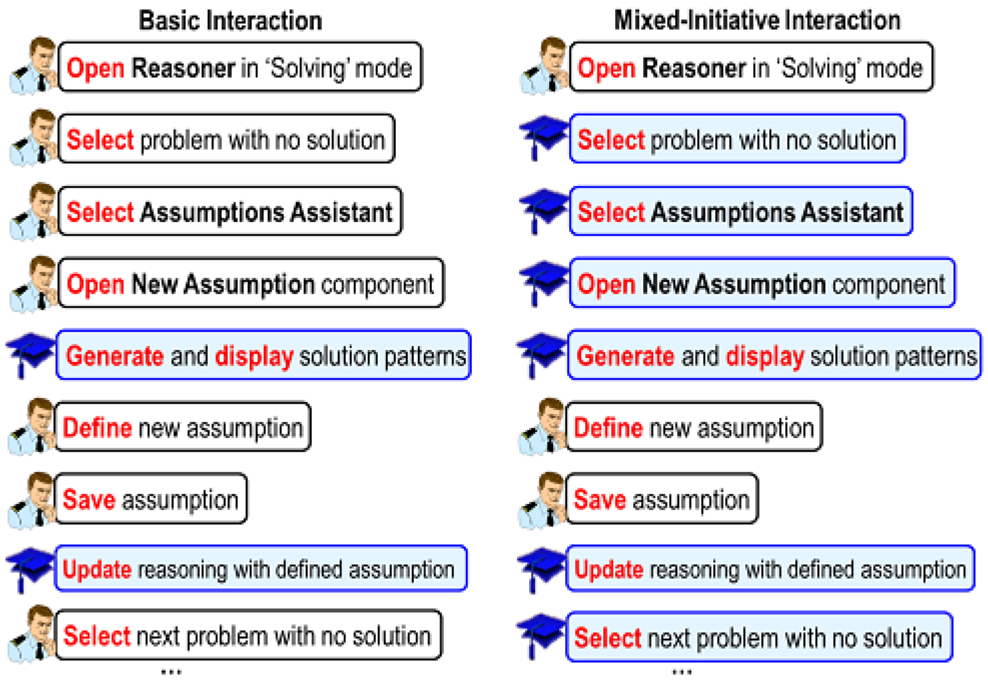

Basic Interaction
Open Reasoner in 'Solving' mode
Select problem with no solution
Select Assumptions Assistant
Open New Assumption component
Generate and display solution patterns
Define new assumption
Save assumption
Update reasoning with defined assumption
Select next problem with no solution
...
Mixed-Initiative Interaction
Open Reasoner in 'Solving' mode
Select problem with no solution
Select Assumptions Assistant
Open New Assumption component
Generate and display solution patterns
Define new assumption
Save assumption
Update reasoning with defined assumption
Select next problem with no solution
...

Figure 11. Basic versus mixed-initiative user interaction with the Assumption Assistant.

8.3 Developing a Mixed-Initiative Interaction Agent

We have developed a systematic approach to building such mixed-initiative interaction agents, in the context of extending an existing multi-agent system with a new mixed-initiative agent. This process, which is illustrated Figure 12 and detailed in (Marcu, 2009), starts with a capability analysis that identifies the high level tasks to be performed with the new agent. For example, in the case of the Assumption Assistant, these tasks include “View assumptions defined for the current problem”, “Define a new assumption” and “Modify a previously defined assumption”.

This is followed by a task analysis step in which a conceptual interaction model is developed for each high level task. Basically, a conceptual interaction model is a plan (or set of plans) that must be executed in order to accomplish a task. During the development of such a conceptual interaction plan, the elementary actions that have to be performed by the agent or by the user to accomplish the task are identified and formally specified. For example, an elementary action for the Assumption Assistant is “Open New Assumption component”, shown in Figure 11, which would invoke the new assumption editing component.

The third step, agent development, has two parts, the implementation of the agent’s functionality and the development of an executable interaction model for the agent. In the traditional approach, these two parts are both software development activities. In our approach, the development of the interaction model was transformed into a knowledge engineering activity.

First, methods to execute the identified elementary actions are implemented in the agent. Next, the conceptual interaction models are translated into state-based interactions. Our approach exploits the complementarity between the advantages and, respectively, the limitations of conceptual interaction models and state-based interaction models by using the models for the steps that are the most useful, as detailed in (Marcu, 2009).

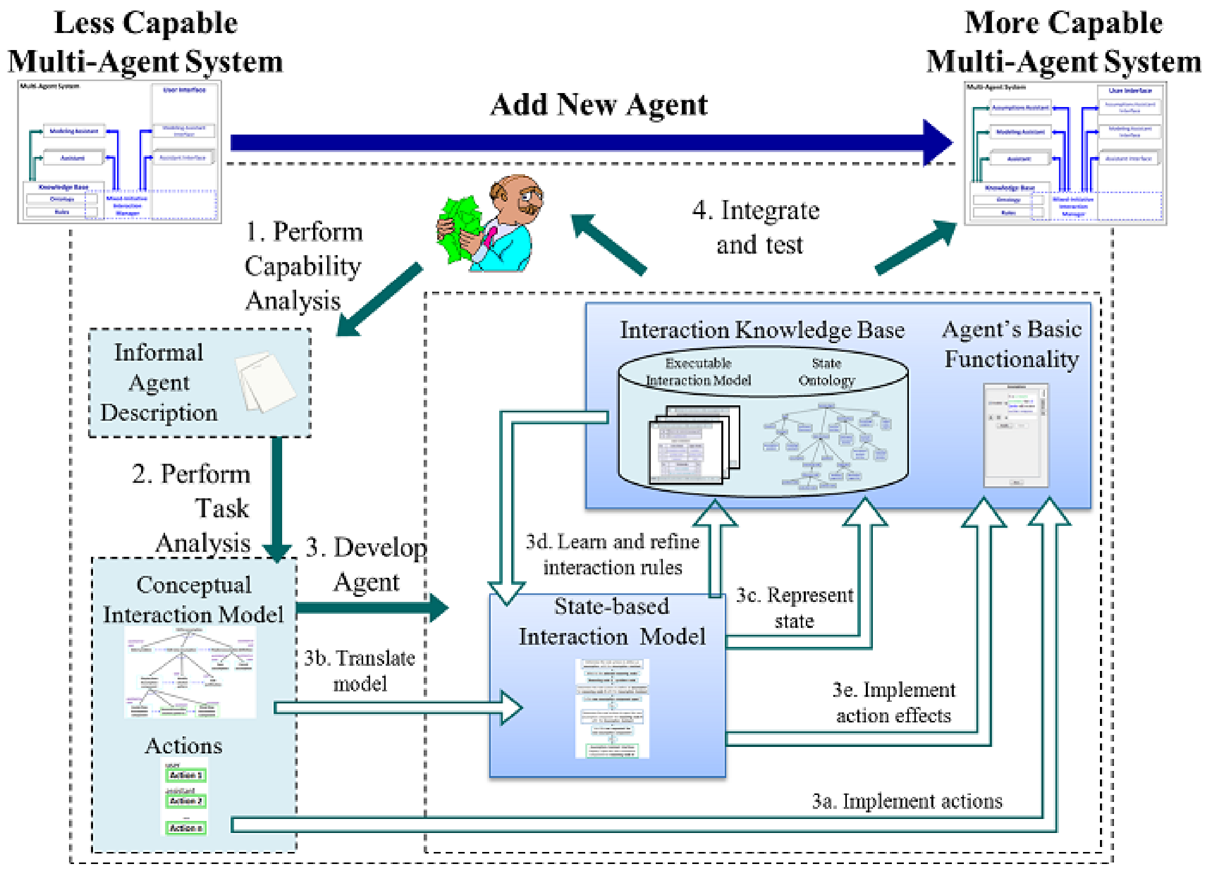

Less Capable Multi-Agent System
More Capable Multi-Agent System
Add New Agent
1. Perform Capability Analysis
4. Integrate and test
Informal Agent Description
2. Perform Task Analysis
Conceptual Interaction Model
Actions
3. Develop Agent
3b. Translate model
Interaction Knowledge Base
Executable Interaction Model
State Ontology
Agent's Basic Functionality
3d. Learn and refine interaction rules
3c. Represent state
State-based Interaction Model
3e. Implement action effects
3a. Implement actions

Figure 12. General approach to the development of a mixed-initiative interaction agent.

We have developed a systematic process for translating conceptual interaction models into their equivalent state-based interaction models by defining a set of translation guidelines, as described in (Marcu, 2009). The translation process also results in the design of a state ontology for representing the state of the interaction, as well as the identification of the effects of each elementary action, which have to be implemented in the agent. The state-based interaction model is then used to teach the agent how to interact with the user by defining examples of interactions from which the agent will learn general interaction rules.

The result is an interaction knowledge base which contains the state ontology and, respectively, the tasks and rules that represent the executable interaction model. The interaction knowledge base, together with the basic functionality, represents the new agent. This agent will interact with the user the way it was taught to interact.

The developed agent can be further refined by using a process similar to the one used for developing it. There are two types of refinements:

- Extend the capability of the agent by adding new actions (which may require updating the basic functionality of the agent through software engineering methods).
- Refine the executable interaction model through knowledge engineering methods.

9. LEARNING AND TUTORING AGENT SHELL

9.1 From Expert System Shells to Learning and Tutoring Agent Shells

A main objective of this research was to enable rapid development and easy maintenance of knowledge-based problem solving and decision-making assistants that can capture, use, preserve, and transfer to other users the military expertise which currently takes years to establish, is lost when military experts separate from service, and is costly to replace. One of its main results is a refinement of the concept of *learning agent shell* and the definition of the new concept of *learning and tutoring agent shell.* The concept of *learning and tutoring agent shell* evolved from the concept of *learning agent shell* (Tecuci 1998) which itself evolved from the concept of *expert system shell* (Clancey, 1984). Its general architecture is represented in Figure 13 and described in more detail in (Le, 2008; Le et al, 2008a).

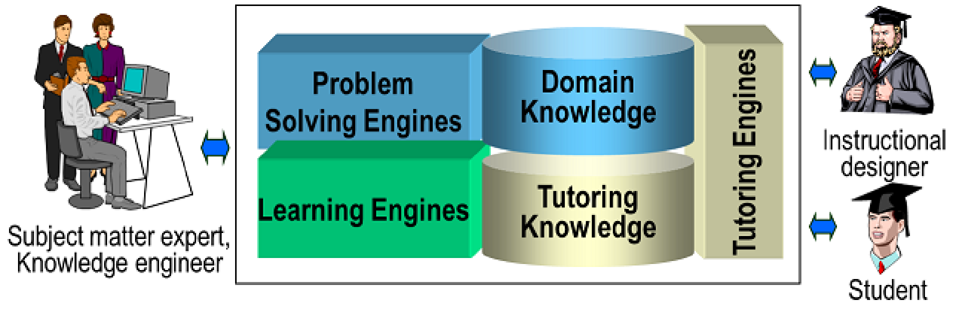


Figure 13. Learning and tutoring agent shell.

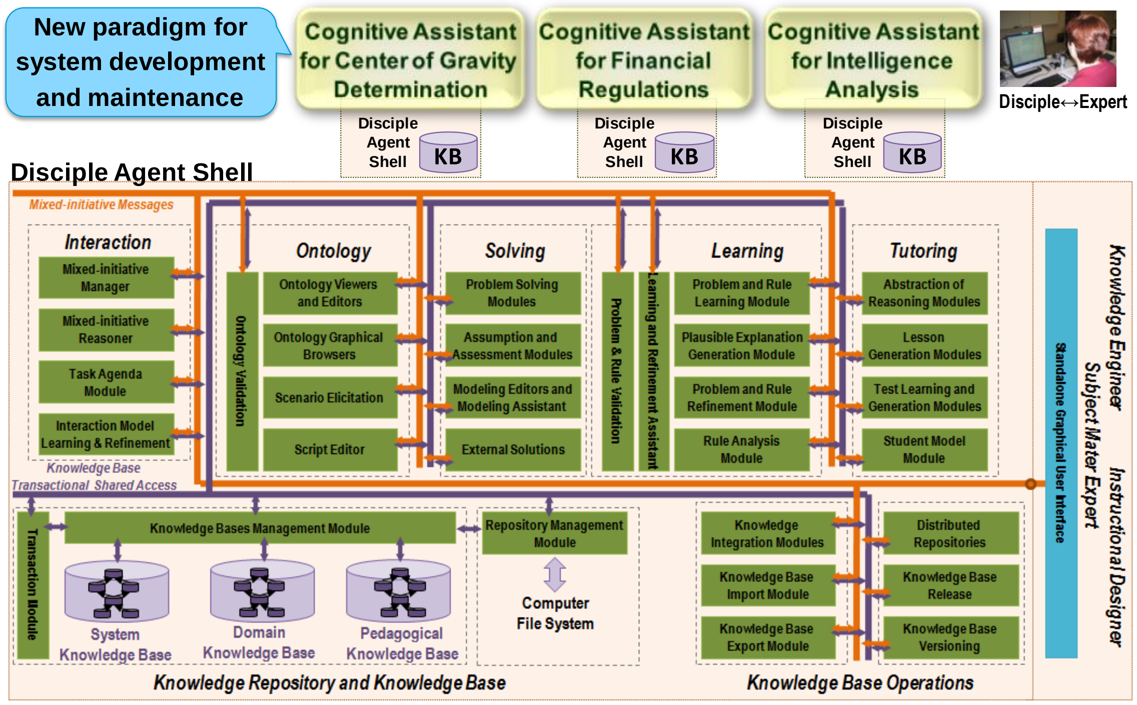
An *expert system shell* is a system that consists of an inference engine for a certain class of tasks (e.g., planning, design, diagnosis, monitoring, prediction, interpretation, etc.) and supports a representation formalism in which a knowledge base can be encoded. If the inference engine of an expert system shell is adequate for a certain expert task (e.g. planning), then the process of building an expert system for that type of tasks is reduced to the building of the knowledge base.

However, even with using an expert system shell, building an expert system remains a very difficult and time-consuming task because of the complexity of acquiring expert knowledge. In response to this challenge, we have developed the concept of learning agent shell, a new type of tool which, in addition to a general problem solving engine, it contains a learning engine and a general knowledge base structured into an object ontology and a set of rules. Building an expert system for a specific application consists in customizing the shell for that application and in developing the knowledge base. The learning engine facilitates the building of the knowledge base by subject matter experts and knowledge engineers. In this project we have refined the concept of learning agent shell to include generic knowledge for evidence-based reasoning from the Science of Evidence, as discussed in Section 3.3. This further facilitates the development of the knowledge base for a new system which already contains generic knowledge for evidence-based reasoning.

A learning and tutoring agent shell extends a learning agent shell with a generic pedagogy knowledge base, tutoring engine, and corresponding interfaces. It is a tool for building learning and tutoring agents which can capture and use subject matter expertise, as well as transfer it to other users.

9.2 Disciple 2009 Shell and Methodology

The developed agent learning theory has been implemented into the Disciple 2009 learning and tutoring agent shell, the architecture of which is shown at the bottom of Figure 14. The Disciple 2009 Shell includes general modules for user-agent interaction, ontology representation, problem solving, learning, tutoring, as well as general knowledge for evidence-based reasoning (both ontology and rules).



New paradigm for system development and maintenance
Cognitive Assistant for Center of Gravity Determination
Cognitive Assistant for Financial Regulations
Cognitive Assistant for Intelligence Analysis
Disciple↔Expert
Disciple Agent Shell
KB
Disciple Agent Shell
Mixed-initiative Messages
Interaction
Mixed-initiative Manager
Mixed-initiative Reasoner
Task Agenda Module
Interaction Model Learning & Refinement
Ontology
Ontology Validation
Ontology Viewers and Editors
Ontology Graphical Browsers
Scenario Elicitation
Script Editor
Solving
Problem Solving Modules
Assumption and Assessment Modules
Modeling Editors and Modeling Assistant
External Solutions
Problem & Rule Validation
Learning and Refinement Assistant
Learning
Problem and Rule Learning Module
Plausible Explanation Generation Module
Problem and Rule Refinement Module
Rule Analysis Module
Tutoring
Abstraction of Reasoning Modules
Lesson Generation Modules
Test Learning and Generation Modules
Student Model Module
Knowledge Base Transactional Shared Access
Transaction Module
Knowledge Bases Management Module
System Knowledge Base
Domain Knowledge Base
Pedagogical Knowledge Base
Repository Management Module
Computer File System
Knowledge Integration Modules
Knowledge Base Import Module
Knowledge Base Export Module
Distributed Repositories
Knowledge Base Release
Knowledge Base Versioning
Knowledge Repository and Knowledge Base
Knowledge Base Operations
Standalone Graphical User Interface
Knowledge Engineer
Subject Mater Expert
Instructional Designer

Figure 14. Disciple shell and Disciple cognitive assistants.

The learning and tutoring agent shell allows a new paradigm for rapid development, as well as easy maintenance and adaptation of knowledge-based cognitive assistants. The top part of Figure 14 shows three Disciple cognitive assistants developed with the Disciple shell. Each assistant incorporates the shell and its development consists in developing the knowledge base. However, in the case of the Cognitive Assistant for Intelligence Analysis (which is the most-recently developed one), this knowledge base already includes general knowledge for evidence-based reasoning from the Science of Evidence. Therefore, what remained to be acquired was the expert knowledge specific to the agent's domain. Moreover, this knowledge was rapidly acquired by employing the developed learning methods. Maintenance and adaptation was done through additional learning.

In the next section we will briefly discuss the cognitive assistants from the top of Figure 14 and the transition of this research.

10. DISCIPLE Cognitive Assistants

Successive versions of the Disciple shell have been used to develop and transition several Disciple cognitive assistants. Their development has been partially supported by several organizations, including the Air Force Research Laboratory (Tecuci and Boicu, 2008a), the Intelligence Community (Tecuci et al., 2008b), the National Science Foundation and Exprentis Inc., and George Mason University.

10.1 Cognitive Assistant for Center of Gravity Analysis

Disciple-COG, for center of gravity analysis, was initially developed with a previous version of the Disciple shell, but using the same learning approach where a subject matter expert has taught it to reason, based on specific examples and explanations. Successive versions of this system have been used, continuously, in elective courses at the Army War College, since 2001. Each year, its knowledge base was easily updated based on students' feedback, this supporting our claim for easy maintenance of the Disciple agents. We have published a textbook to support the continuous use of Disciple-COG in US Army War College courses (Tecuci et al., 2008c).

10.2 Cognitive Assistant for Intelligence Analysis

We have developed successive versions of a cognitive assistant for intelligence analysis (Tecuci et al., 2008b; Schum et al., 2009a). They have been used in several elective courses at the US Army War College, in several short courses for other organizations, and in several experiments with intelligence analysts.

In Fall 2008, ODNI has challenged the attendees of the Open Source Intelligence Conference to answer the following question in at most 9 days: "According to open sources, who will be the global leader in alternative fuels and why?" We have used this opportunity to show what can be accomplished with the Disciple shell in at most one day, on a type of problem on which it has not been previously trained. We have developed the top-level logic for assessing whether United States will be the global leader in wind power within the next 10 years. This allowed us to teach Disciple the corresponding reasoning rules. Disciple was able to use its general knowledge of evidence-based reasoning to develop the bottom-level logic for assessing the believability of evidence. The reasoning learned from the analysis of the United States and wind power was automatically applied by the Disciple agent to analyze other countries (e.g., United Kingdom, China, etc.) and other alternative fuels (e.g., solar power).

10.3 Cognitive Assistant for Financial Regulations

With support from the National Science Foundation's Small Business Technology Transfer Phase I and Phase II Program, we have developed Disciple-FS, a customized version of the Disciple shell for the development, utilization, and maintenance of regulatory knowledge bases in the financial services industry. Exprentis (http://www.exprentis.com/) has used Disciple-FS to develop a regulatory knowledge base for brokerage and trading compliance and for anti-money laundering compliance. The knowledge base includes a financial ontology and a set of reasoning rules which have been learned from specific problem solving episodes specified by a regulatory compliance expert.

10.4 Other Disciple Cognitive Assistants

We have developed other prototype Disciple cognitive assistants for various types of problems, including evaluating a PhD advisor, evaluating the believability of a website, evaluating teaching practices, and medical triage.

11. CONCLUSIONS AND FUTURE RESEARCH

We conclude with some very promising directions that are emerging from this research. In Section 2.1 we have discussed a computational approach to evidence-based reasoning and learning, which holds high promise, but it is only partially developed. We plan to develop the "evidence in search of hypothesis part" of this computational framework. This involves the ability to learn abductive rules and to use them in assisting the analyst in performing insightful reasoning.

The reasoning approach of a Disciple agent shows a natural integration of logic and probability for evidence-based reasoning. However, as shown in (Schum, 2001a), none of the exiting probability views can independently deal with all the characteristics of evidence (i.e., incompleteness, inconclusiveness, ambiguity, dissonance, and various levels of believability), although they can deal with these characteristics collectively. Thus, a very promising research direction is to develop an approach to evidence-based reasoning which is hybrid in two senses: in integrates logic and probability, and it also integrates different views of probability. In particular, we plan to develop an approach for learning of solution synthesis rules from subject matter experts, and for reasoning with these rules, which will be based on different probability views.

Finally, we would like to further evolve the concept of agent shell to provide an integrated approach to the design, development and use of adaptive mixed-initiative knowledge-based systems for problem-solving and decision-making in uncertain and dynamic environments, systems which support collaboration and information sharing, and are based on the notion of valued information (i.e., relevant, believable) at the right time (Hayes-Roth, 2006).

12. CONTRIBUTORS AND PUBLICATIONS

The following professors, researchers, PhD and MS students have been supported by this AFOSR grant: Gheorghe Tecuci, Mihai Boicu, Cristina Boicu, Dorin Marcu, Vu Le, Israa Taha, and Salman Jamali.

Twenty one papers have been published during this project which has provided partial support:

- 1 book (Tecuci et al., 2008c)
- 2 PhD theses (Le, 2008; Marcu 2009)
- 1 journal special issue (Tecuci, Boicu, Cox, 2007a)
- 7 journal papers (Schum et al., 2009a; Tecuci et al., 2007b; Tecuci et al., 2007c; Tecuci et al., 2007d; Tecuci et al., 2008a; Tecuci et al., 2008b; Tecuci et al., 2010)
- 6 conference or workshop papers (Boicu C et al., 2007; Boicu M et al., 2008a; Boicu M et al., 2008b; Le et al, 2008a; Le et al., 2008b; Schum et al., 2009b)
- 4 research reports (Tecuci and Boicu, 2008a; Tecuci and Boicu, 2008b; Tecuci and Boicu, 2008c; Tecuci et al., 2009)

13. REFERENCES

Ahn A.C., Tewari M., Poon C.S., Phillips R.S., The clinical applications of a systems approach. *PLoS Med* 3(7): e209. DOI: 10.1371/journal. pmed.0030209, 2006.

Anderson, T., Schum, D., Twining, W., *Analysis of Evidence* (2nd Edition). Cambridge University Press, 2005.

Arnold, B., Korsan, L., Henrion, M., Mulford, R., *Analytica Tutorial*, http://analyticaonline.com/ana/Tutorial4_0_0.pdf, 2007.

Boicu M., Modeling and Learning with Incomplete Knowledge, *PhD Thesis in Information Technology*, Learning Agents Laboratory, School of Information Technology and Engineering, George Mason University, 2002.

Boicu C., Tecuci G., Boicu M., Learning Complex Problem Solving Expertise from Failures, in *Proceedings of the Sixth International Conference on Machine Learning and Applications* (ICMLA'07), pp.118-123, Cincinnati, Ohio, December 13-15, 2007.

Boicu M., Tecuci G., Marcu D., Mixed-Initiative Assistant for Modeling Expert's Reasoning, *Proceedings of the AAAI-05 Fall Symposium on Mixed-Initiative Problem-Solving Assistants,* Arlington, Virginia, November 4-6, 2005.

Boicu M., Le V., Marcu D., Boicu C., Barbulescu M., Tecuci G., An ITS-Authoring Shell and an Authored ITS for an Ill-Structured Domain, in *Proceeding of the Demonstration Program of the 9th International Conference on Intelligent Tutoring Systems*, 2008a.

Boicu M., Tecuci G., Schum D., Intelligence Analysis Ontology for Cognitive Assistants, in *Proceedings of the Conference "Ontology for the Intelligence Community: Towards Effective Exploitation and Integration of Intelligence Resources,"* George Mason University, Fairfax, Virginia Campus, 3-4 December 2008b.

Bresina, J.L., Morris, P.H., Mixed-Initiative Planning in Space Mission Operations, *AI Magazine*, 28:2, 2007.

Buchanan, B.G., and Wilkins, D.C. (eds.), *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems,* Morgan Kaufmann, San Mateo, CA, 1993.

Cheesman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., Freeman, D., AUTOCLASS: A Bayesian Classification System. *Proceedings of AAAI 1988*, pp 607-611, 1988.

Clancey, W.J., NEOMYCIN: Reconfiguring a rule-based system with application to teaching. In: Clancey, W.J., Shortliffe, E.H. (eds.) Readings in Medical Artificial Intelligence, pp.361-381. Reading, MA: Addison-Wesley, 1984.

Domingos, P., Richardson, M., Markov Logic Networks, *Machine Learning*, 62, 107-136, 2006.

Eco, U., Horns, Hooves, Insteps: Some Hypotheses on Three Types of Abduction, in Eco, U. and Sebeok, T. (eds.), *The Sign of Three: Dupin, Holmes, Peirce*, pp. 198-220, Indiana University Press: Bloomington, IN, 1983.

Ferguson, G., Allen, J., Mixed-Initiative Systems for Collaborative Problem Solving, *AI Magazine*, 28:2, 2007.

Ghallab M., Nau D., and Traverso P., *Automatic Planning:Theory and Practice,* Morgan Kaufmann, 2004.

Hayes-Roth, F., Model-based Communication Networks and VIRT: Orders of Magnitude Better for Information Superiority. Presented at MILCOM-2006, Washington, D.C., 2006.

Kadie, C.M., Hovel, D., Horvitz, E., MSBNx: A Component-Centric Toolkit for Modeling and Inference with Bayesian Networks, *Microsoft Research Technical Report MSR-TR-2001-67*, http://research.microsoft.com/adapt/MSBNx/default.aspx, July 2001.

Kent S., ''Words of Estimated Probability,'' in Donald P. Steury, ed., *Sherman Kent and the Board of National Estimates: Collected Essays* (Washington, DC: CIA, Center for the Study of Intelligence, 1994).

Le V., Abstraction of Reasoning for Problem Solving and Tutoring Assistants, *Ph.D. Dissertation in Information Technology*, Learning Agents Center, Volgenau School of IT&E, George Mason

University, Spring 2008. Spring 2008. Available from the George Mason University library (http://library.gmu.edu/) and UMI/ProQuest (http://www.proquest.com/).

Le V., Tecuci G., Boicu M., Agent Shell for the Development of Tutoring Systems for Expert Problem Solving Knowledge, in *Proceedings of the 9th International Conference on Intelligent Tutoring Systems (ITS2008)*, LNCS 5091, pp. 228–238, Springer-Verlag, Berlin Heidelberg / Montreal, 2008a.

Le V., Tecuci G., Boicu M., A New Approach to Rapid Construction of Knowledge-Based Tutoring Systems, in *Proceeding of the 2008 Technology Enhanced Learning Conference*, 8 pages, Hanoi, Vietnam, December 2008b.

Lemmer, J., Java Causal Analysis Tool, *AFRL/IF Research Report,* 2007.

Lenat D., CYC: A large-scale investment in knowledge infrastructure, *Communications of the ACM*, 38:11, pp 33–38, 1995.

Marcu D., Learning of Mixed-Initiative Human-Computer Interaction Models. Ph.D. Dissertation in Computer Science. Learning Agents Center, Volgenau School of IT&E, George Mason University, January 2009. Available from the George Mason University library (http://library.gmu.edu/) and UMI/ProQuest (http://www.proquest.com/).

Milch, B., Marthi, B., Russell, S., Sontag, D., Ong, D., Kolobov., A., BLOG: Probabilistic Models with Unknown Objects, *Proceedings 19th International Joint Conference on Artificial Intelligence* (IJCAI), pp. 1352-1359, 2005.

Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubezy, M., Eriksson, H., Noy, N. F., Tu, S. W., The Evolution of Protégé: An Environment for Knowledge-Based Systems Development, *International Journal of Human-Computer Interaction*, 58:1, pp 89-123, http://smi.stanford.edu/pubs/SMI_Abstracts/SMI-2002-0943.html, 2003.

Myers, K., Berry, P., Blythe, J., Conley, K., Gervasio, G., McGuinness, D., Morley, D., Pfeffer, A., Pollack, M., Tambe, M., An Intelligent Personal Assistant for Task and Time Management, *AI Magazine*, 28:2, 2007.

Nilsson, N.J., *Problem Solving Methods in Artificial Intelligence,* NY: McGraw-Hill, 1971.

Nodelman, U., Koller, D., Shelton, C.R., Expectation Propagation for Continuous Time Bayesian Networks, *Proceedings of the Twenty-first Conference on Uncertainty in AI* (UAI), pp 431-440, 2005.

Santos, E. Jr., Dinh, H.T., Consistency of Test Cases in Validation of Bayesian Knowledge Bases, *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, pp 468-475, Boca Raton, FL, 2004.

Schum D.A., *Evidence and Inference for the Intelligence Analyst* (2 Vols), University Press of America, MD: Lanham, 1987.

Schum D.A., *The Evidential Foundations of Probabilistic Reasoning*, Northwestern University Press, 2001a.

Schum D.A., Species of Abductive Reasoning in Fact Investigation in Law, *Cardozo law Review*, pp.1645-1681, July 2001b.

Schum D., Tecuci G., Boicu M., Analyzing Evidence and Its Chain of Custody: A Mixed-Initiative Computational Approach, *International Journal of Intelligence and CounterIntelligence*, Volume 22, Issue 2, pp. 298-319, 2009a.

Schum, D., Tecuci, G., Boicu, M., Marcu, D., Substance-Blind Classification of Evidence for Intelligence Analysis, in *Proceedings of the Conference "Ontology for the Intelligence Community,"* George Mason University, Fairfax, Virginia Campus, 20-22 October 2009b.

Shirazi, A., Amir, E., Probabilistic Modal Logic, *22nd National Conf. on Artificial Intelligence* (AAAI 07), 2007.

Sutton, C., McCallum, A., Rohanimanesh, K., Dynamic Conditional Random Fields, *Journal of Machine Learning Research* (JMLR), Vol. 8 (Mar), pp 693-723, 2007.

Tecuci G., *Disciple: A Theory, Methodology and System for Learning Expert Knowledge*, Thèse de Docteur en Science (in English), University of Paris-South, France, 1988.

Tecuci, G., *Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies*, Academic Press, 1998. http://lac.gmu.edu/publications/1998/TecuciG_Building_Intelligent_Agents/default.htm

Tecuci G., Boicu M., Intelligent Assistants for Distributed Knowledge Acquisition, Integration, Validation, and Maintenance, *Final Report for the AFRL Grant # FA8750-04-1-0257*, 36 pages, Learning Agents Center, Fairfax, VA 22030, April 4, 2008a.

Tecuci G., Boicu M., A Guide for Ontology Development with Disciple, *Research Report 3*, 38 pages, Learning Agents Center, August 22, 2008b.

Tecuci G., Boicu M., Learning-Based Knowledge Representation, *Research Report 4*, 26 pages, Learning Agents Center, September 8, 2008c.

Tecuci G. and Kodratoff Y. (eds.), *Machine Learning and Knowledge Acquisition: Integrated Approaches,* Academic Press, 1995.

Tecuci G., Boicu M., Cox M.T., (Guest editors) *AI Magazine,* Special Issue on Mixed-Initiative Assistants, Volume 28, Number 2, Summer 2007a. http://www.aaai.org/Library/Magazine/vol28.php#Summer

Tecuci G., Boicu M., Cox M.T., Seven Aspects of Mixed-Initiative Reasoning: An Introduction to the Special Issue on Mixed-Initiative Assistants, *AI Magazine*, Volume 28, Number 2, pp. 11-18, Summer 2007b. http://www.aaai.org/Library/Magazine/Vol28/aimag28-02-002.php

Tecuci G., Boicu M., Marcu D., Boicu C. M., Barbulescu, Ayers C., Cammons C., Cognitive Assistants for Analysts, *Journal of Intelligence Community Research and Development*, 2007c.

Tecuci G., Marcu D., Boicu M., Le V., Mixed-Initiative Assumption-Based Reasoning for Complex Decision-Making, *Studies in Informatics and Control,* Volume 16, Number 4, 2007d.

Tecuci G., Boicu M., Marcu D., Barbulescu M., Boicu C., Le V., Hajduk, Teaching Virtual Experts for Multi-Domain Collaborative Planning, *Journal of Software,* Vol. 3, No. 3, pp. 38-59, March 2008a (journal online at http://www.academypublisher.com/jsw/).

Tecuci G., Boicu M., Marcu D., Boicu C., Barbulescu M., Disciple-LTA: Learning, Tutoring and Analytic Assistance, *Journal of Intelligence Community Research and Development (JICRD)*, 10 pages, July 2008b.

Tecuci G., Boicu M., and Comello J. (with contributions from Marcu D., Boicu C., Barbulescu M., Le V., Cleckner W.), *Agent-Assisted Center of Gravity Analysis*, CD with Disciple-COG and Lecture Notes used in courses at the US Army War College and Air War College, GMU Press, ISBN 978-0-615-23812-8, 2008c.

Tecuci, G., Boicu, M., Schum, D., Marcu, D., Overcoming Intelligence Analysis Complexity with Cognitive Assistants, *Research Report 7*, 45 pages, Learning Agents Center, August 2009, updated October 2009.

Tecuci, G., Schum, D., Boicu, M., Marcu, D., Hamilton, B., Wible, B., Teaching Intelligence Analysis with TIACRITIS, submitted to the American Intelligence Journal, March 2010.

Thagard, P.R., *Computational Philosophy of Science,* MIT Press: Cambridge, MA, 1993.

Toffler, A., Science and Change, Forward to Ilya Prigogine and Isabelle Stengers. *Order Out of Chaos: Man's New Dialogue with Nature*. Bantam Books, 1984.

Toulmin, S. E., *The Uses of Argument*, Cambridge University Press, 1963.

Weiss C., ''Communicating Uncertainty in Intelligence and Other Professions,'' *International Journal of Intelligence and CounterIntelligence,* Vol. 21, No. 1, Spring 2008, pp 57–85.

Wigmore, J.H., *The Science of Judicial Proof*, Little, Brown & Co., Boston, MA, 1937.