Advances in Cognitive Systems, Vol 8, 2019.
Also in *Proceedings of the Seventh Annual Conference on Advances in Cognitive Systems*, Technical Report Number COLAB^2 -TR-4, pp.183-204, August 2-5, 2019, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Instructable Cognitive Agents for Autonomous Evidence-Based Reasoning

Gheorghe Tecuci TECUCI@GMU.EDU
Steven Meckl SMECKL@GMU.EDU
Dorin Marcu DMARCU@GMU.EDU
Mihai Boicu MBOICU@GMU.EDU
Learning Agents Center, George Mason University, Fairfax, VA 22030 USA

Abstract

This paper presents a general approach to the development of instructable cognitive agents for automated evidence-based reasoning. This is based on a framework, grounded in the scientific method, for generating and testing hypotheses that explain events of interest, and on an instructable agent shell that implements this framework. The agent shell is customized into two different systems, one for automated intelligence, surveillance, and reconnaissance and the other for cybersecurity. A domain expert teaches the agent through explained examples of investigations of alerts. The agent learns by generalizing these examples, being able to autonomously conduct similar investigations, as demonstrated in an experiment in the area of cybersecurity.

1. Evidence-Based Reasoning

Evidence is any observable sign, datum, or item of information that is relevant in deciding whether a statement or hypothesis (e.g., a scientific or medical claim) is true or false. Throughout history, some of the greatest minds, including Aristotle (384BC–322BC), Galileo Galilei (1564–1642), Isaac Newton (1642–1727), John Locke (1632–1704), William Whewell (1794–1866), Charles Peirce (1839–1914), John Wigmore (1863–1943), and David Schum (1932–2018), have tried to understand and reason about the world through a process of discovery and testing of hypotheses based on evidence. We call this process *evidence-based reasoning*.

Evidence-based reasoning is at the basis of many problem solving and decision-making tasks in a wide variety of domains, including physics, chemistry, history, archaeology, medicine, law, forensics, intelligence analysis, cybersecurity, and many others. For example, in medicine, based on the patient's complaints, a doctor generates possible diagnoses (hypotheses) that would explain them. She performs various medical tests that provide further evidence for or against the various hypothesized illnesses and, based on the obtained evidence, she determines the most likely illness. In forensics, observations at the site of an explosion in a power plant lead to the formulation of several possible causes. Analysis of each possible cause leads to the discovery of new evidence that eliminates or refines some of the causes, and may even suggest new ones. This cycle continues until enough evidence is found to determine the most likely cause. In science, a scientist imagines possible explanatory hypotheses of an unusual phenomenon. Then experiments are designed and performed that provide evidence to test the hypotheses. In law, an attorney makes observations in a criminal case and seeks to generate hypotheses in the form of charges that seem possible in

explaining these observations. Then, for the charge that appears to be justified, attempts are made to deduce further evidence to prove it.

Many books and papers have been written on the obvious complexity of such tasks (Schum, 2001a). Each of them involves drawing defensible and persuasive conclusions from masses of information of all kinds that come from a variety of different sources. The evidence upon which conclusions eventually rest has five major characteristics that make these conclusions necessarily probabilistic in nature. The evidence is always *incomplete* no matter how much we have. It is commonly *inconclusive* in the sense that it is consistent with the truth of more than one hypothesis. Further, the evidence is frequently *ambiguous* and, in most situations, *dissonant*, some of it favoring one hypothesis while other evidence favoring other hypotheses. Finally, the evidence comes from sources having different levels of credibility. Arguments to test the considered hypotheses are necessary in order to establish and defend the three major credentials of evidence: its relevance, its credibility, and its inferential force or weight. These arguments, often stunningly complex, rest upon both *imaginative* and *critical reasoning*.

Our objective is to develop cognitive agents that can autonomously perform such complex tasks in a transparent manner. Such agents would have many and very useful applications. Consider, for example, the automatic monitoring of an industrial installation, such as a nuclear plant, where these agents continuously investigate any sign of potential abnormal activity. One could also imagine an evidence-based reasoning approach to the permanent monitoring of our personal health status based on temperature, heart rate, and other health indicators from personal devices such as smart watches and fitness bands.

Over the past several years we have focused on two different EBR domains, *Intelligence, Surveillance, and Reconnaissance (ISR)* and *Cybersecurity*, developing two different systems that have led to a unifying computational EBR framework and agent architecture that may now be further generalized and applied to other domains. The first system is CAPIP, Cognitive Agent for Persistent Intelligence Processing, which reasons with evidence from MITRE's Integrated Environment for Persistent Intelligence. CAPIP's objective is to continuously monitor and understand the situation in a certain area of the world, predict the behavior and intent of the entities of interest, and identify threats. The second system is CAAPT, Cognitive Agent for Advanced Persistent Threats. This is integrated into a Cybersecurity Operations Center and automatically investigates cybersecurity alerts that may be caused by intrusions into our networks.

We will first present the computational approach to evidence-based reasoning, which is at the basis of these agents, illustrating it with a CAPIP maritime ISR example. Next we present the architecture of an agent shell that can be instructed by a subject matter expert to perform evidence-based reasoning. After this, we describe the use of this shell to develop CAAPT, as well as the results of an experiment of incrementally training CAAPT to detect advanced persistent threats.

2. Computational Approach to Evidence-Based Reasoning

Developed in the framework of the scientific method, the computational approach to evidence-based reasoning (EBR) views this process as *ceaseless discovery of evidence, hypotheses, and arguments* in a non-stationary world, involving collaborative computational processes of *evidence in search of hypotheses*, *hypotheses in search of evidence*, and *evidentiary testing of hypotheses* (see Figure 1). We will describe it in the context of a maritime ISR application where ships are monitored through an *Automatic Identification System* (AIS) that, according to the International Maritime Organization (2019), is designed to automatically provide information about a ship (e.g., the ship's name, course and speed, classification, call sign, registration number) to other ships and to coastal authorities.

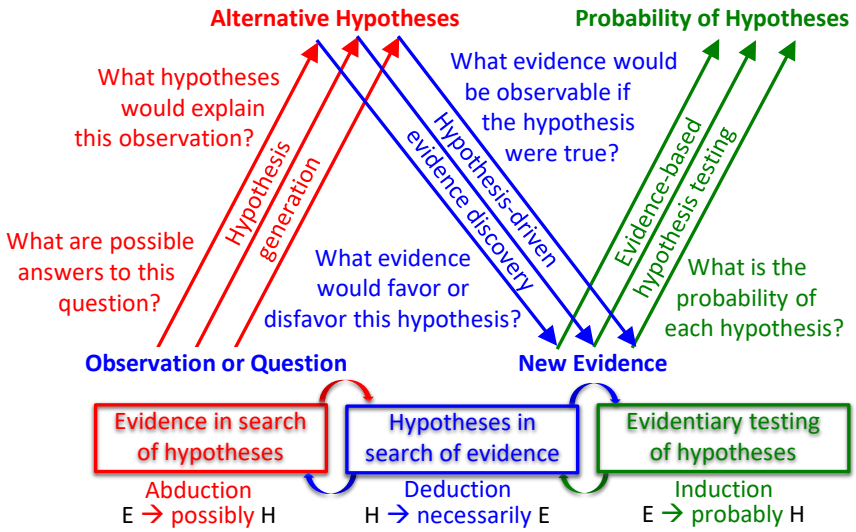

Figure 1. The main processes of evidence-based reasoning.

2.1 Evidence in Search of Hypotheses

Let us assume that an agent monitoring the AIS data generates the alert that the AIS tracking signal for Ship1 was lost sometime between Time1 and Time 2 near Location1. The last signal received was at Time 1 and, at Time 2, when tracking was checked again, there was no signal. The question is: *What hypotheses would explain this unusual event?* Through *abductive* (*imaginative*) reasoning, which shows that something is *possibly* true (Peirce, 1955; Eco, 1983; Thagard, 1993; Schum, 2001b), the EBR agent generates the set of *alternative hypotheses* from Figure 2 that may explain this observation.

Because of the evidence E1 from the agent monitoring the AIS data, it is possible that the event described by E1actually happened. It is further possible that the AIS equipment of Ship1 was turned off intentionally by its crew, and this was done because Ship1 performed covert goods transfer with another ship (see the abductive inferences on the left side of Figure 2). Should the agent conclude that the covert goods transfer actually happens? No, because there are many other hypotheses that may explain the alert. For example, it is also possible that the AIS equipment was turned off intentionally by other individuals, or that it was damaged (see the middle part of this figure). Further up in Figure 2, if the AIS equipment of Ship1 was turned off intentionally by its crew, then it is indeed possible that Ship1 performs covert goods transfer, but it is also possible that it performs illegal fishing or that it tries to avoid tracking by pirates. These top-level hypotheses are the hypotheses of interest that need to be investigated to determine which of them is true. For this, however, one needs to discover more evidence, as discussed in the next section.

In some domains, particularly in intelligence analysis, the starting point of the EBR process may not be an observation, but an intelligence question whose possible answers are the hypotheses to be investigated, as depicted in the left side of Figure 1.

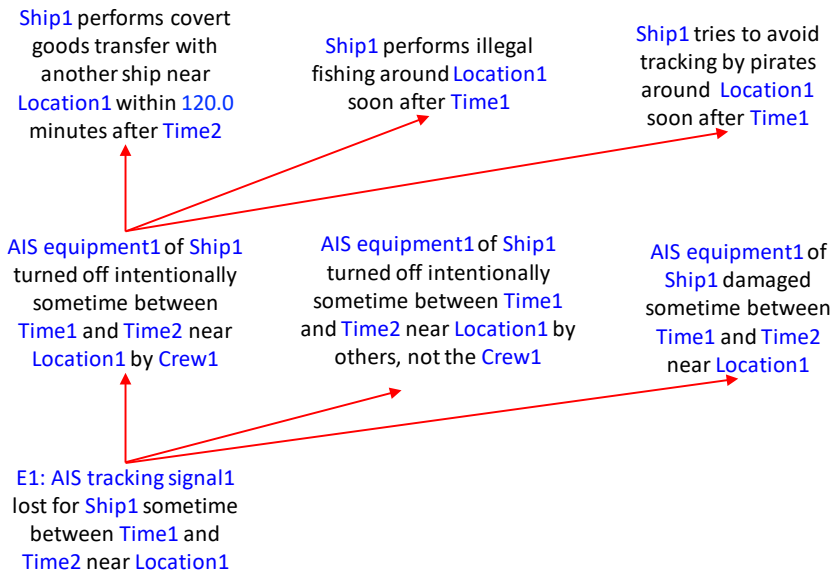

Figure 2. Evidence in search of hypotheses.

2.2 Hypotheses in Search of Evidence

To discover new evidence, the agent puts each of the generated hypothesis to work, guiding the collection of relevant evidence. The question is, *What evidence would be observable if this hypothesis were true?* Through *deductive* reasoning, which shows that something is *necessarily* true, it decomposes the hypothesis into simpler and simpler hypotheses, and uses the simplest hypotheses to generate new lines of inquiry and discover new evidence. The reasoning might go as follows: If H were true then the sub-hypotheses H_1, H_2, and H_3 would also need to be true. But if H_1 were true then one would need to observe evidence E_1, and so on (see the blue, middle side, of Figure 1). This process leads to the discovery of new evidence.

A broader question that guides the discovery of evidence is, *What evidence would favor or disfavor this hypothesis?* Figure 3 illustrates this hypothesis decomposition process: One could infer that the AIS equipment of Ship1 was turned off intentionally by its crew by arguing that Ship1 was operating normally, that the AIS reception was working, and that the AIS tracking signal was lost for Ship1. Further down, one can argue that Ship1 was operating normally by arguing that no distress signal was received from it. To support this, one would collect evidence from the AIS Status Checker, and so on.

The decomposition tree from Figure 3 is a favoring argument for the top hypothesis and that is why it is under the left (green) square. There may be additional favoring arguments, as well as disfavoring arguments, the latter ones under the right (red) square. These arguments will end in evidence collection requests that will return evidence to test the top hypothesis, as discussed next.

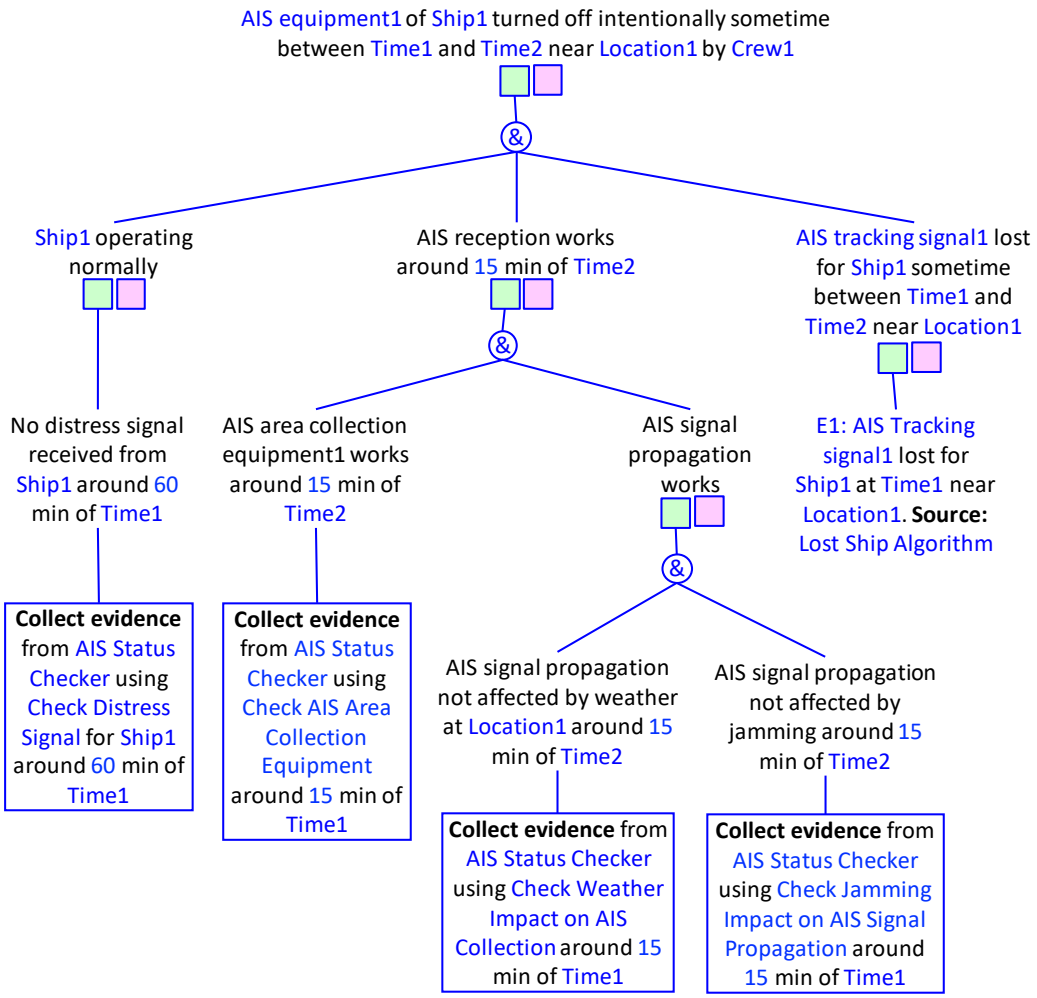

Figure 3. Hypothesis in search of evidence.

2.3 Evidentiary Testing of Hypotheses

The agent uses the discovered evidence to test the generated alternative hypotheses from Figure 2. The tree structure in Figure 3 is a type of Wigmorean probabilistic inference network where the probabilities of the bottom hypotheses are assessed based on the collected evidence, and the probabilities of the upper level hypotheses are assessed based on the probabilities of their subhypotheses (Wigmore 1913; Schum, 2001a; Tecuci et al., 2016a). These Wigmorean networks naturally integrate logic and Baconian probability (Cohen, 1977; 1989) with Fuzzy qualifiers (Zadeh, 1983), such as "barely likely," "likely," or "almost certain," (see Table 1), being able to deal with all the five characteristics of evidence, namely incompleteness, inconclusiveness, ambiguity, dissonance, and credibility level (Schum, 2001a; Tecuci et al., 2016b, pp. 159-172). This integrated logic and probability system uses the min/max probability combination rules common to the Baconian and the Fuzzy probability views. These rules are much simpler than the Bayesian probability combination rule, which is important for the human understandability of the

Table 1. Probability scale.

L11	100%	certain
L10	95-99%	almost certain
L09	90-95%	very likely+
L08	85-90%	very likely
L07	80-85%	very likely-
L06	75-80%	more than likely+
L05	70-75%	more than likely
L04	65-70%	likely+
L03	60-65%	likely
L02	55-60%	likely-
L01	50-55%	barely likely
L00	0-50%	lacking support

analysis. One can directly assess the probability of a hypothesis based on an item of evidence by assessing the three credentials of evidence: credibility, relevance, and inferential force, as shown in Figure 4. The *credibility* of evidence answers the question, *"Which is the probability that the evidence is true?"* The source of E2 in the figure is the collection agent "AIS Status Checker using Check Distress Signal," with known accuracy of 85-90% (very likely) associated with this sensor. The *relevance* of evidence to a hypothesis answers the question, *"What would be the probability of the hypothesis if the evidence were true?"* In this case, if E2 is true then H is true, and therefore the relevance is certain.

The *inferential force or weight* of the evidence on the hypothesis answers the question, *"What is the probability of the hypothesis, based only on this evidence?"* Obviously, an irrelevant item of evidence will have no inferential force, and will not convince us that the hypothesis is true. An item of evidence that is not credible will have no inferential force either. Only an item of evidence that is both very relevant and very credible may convince us that the hypothesis is true. Consistent with both the Baconian and the Fuzzy min/max probability combination rules, the inferential force of an

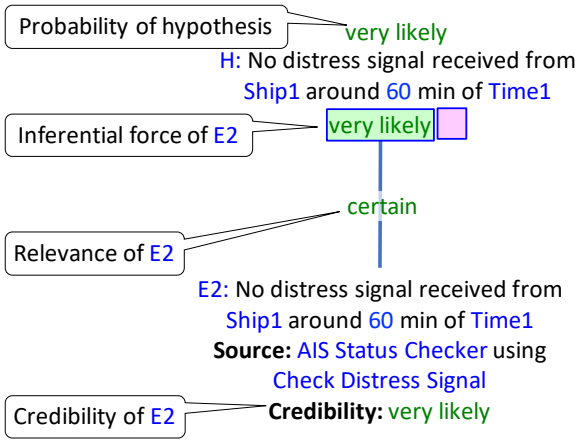

Figure 4. The credentials of evidence.

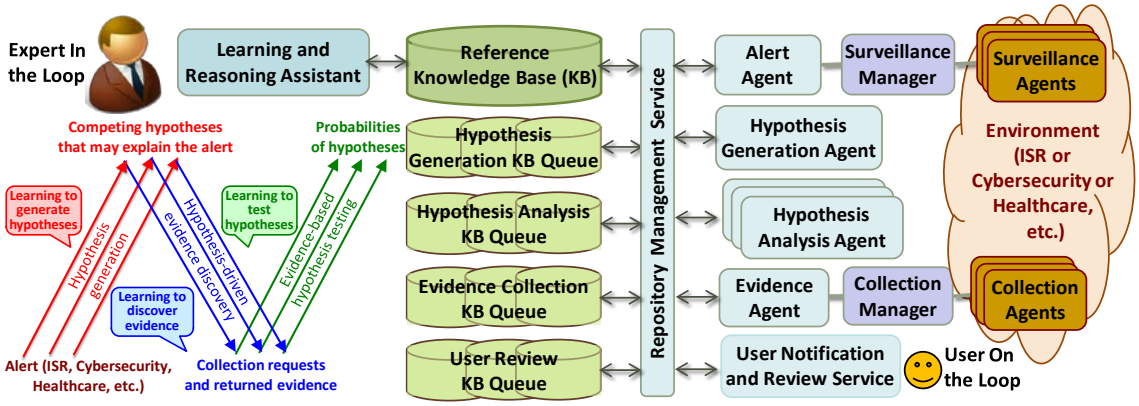

Figure 5. Instructable agent shell for evidence-based reasoning.

item of evidence on a hypothesis is determined as the minimum between its credibility and its relevance which, in this illustration is very likely.

Because in the situation from Figure 4 we have only one item of favoring evidence, its inferential force on the hypothesis is also the probability of the hypothesis. In general, however, the probability of the hypothesis would be the result of balance of probabilities between the combined inferential force of the favoring evidence items and the combined inferential force of the disfavoring items. Notice that, as shown at the bottom of Figure 1, EBR is a recursive process where, for example, the discovery of new evidence may lead to the generation of new hypotheses or the modification of the existing ones which, in turn, may lead to the discovery of new evidence.

3. Instructable Agent Shell

This computational approach to EBR is the basis of an instructable agent shell whose architecture is shown in Figure 5. Its two main components are a mixed-initiative *Learning and Reasoning Assistant* and an autonomous *Multi-Agent Reasoner*. We describe both components in this section.

3.1 Learning and Reasoning Assistant

As illustrated in the left-hand side of Figure 5, the subject matter expert teaches the Learning and Reasoning Assistant how to conduct an investigation of an alert, by following the EBR process discussed in Section 2. As a result, the agent learns to generate hypotheses from alerts, to discover relevant evidence, and to test the hypotheses based on the discovered evidence.

The Learning and Reasoning Assistant employs a hybrid knowledge representation consisting of an ontology and various types of rules. However, even before being taught, the Learning and Reasoning Assistant already has a significant amount of general knowledge from the *science of evidence*. This includes an ontology of evidence with different types of tangible and testimonial evidence (e.g., real tangible evidence, demonstrative tangible evidence, unequivocal testimonial evidence, equivocal testimonial evidence, unequivocal testimonial evidence based upon direct observation, unequivocal testimonial evidence obtained at second hand). It also includes reasoning rules for assessing the credibility of the various types of evidence. For example, to assess the credibility of an item of unequivocal testimonial evidence based upon direct observation one needs to assess the competence, the veracity, and the accuracy of the source of this evidence item. These

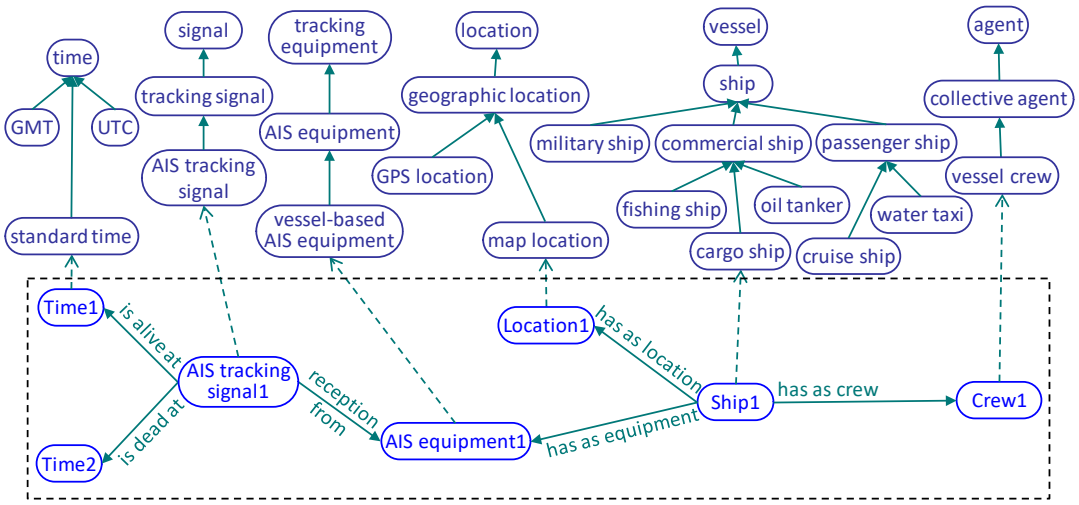

Figure 6. Ontology fragment with example explanation.

credibility credentials are further decomposed into simpler credibility credentials. For example, to assess the accuracy of the source one needs to assess the source's observational accuracy and the source's objectivity, and so on, down to the level of elementary credentials that need to be assessed based on other items of evidence.

In addition to its domain-independent knowledge, the agent is provided with a domain (e.g., ISR or cybersecurity) ontology consisting of a hierarchy of concepts and instances in that domain, together with their properties and relationships, as illustrated in Figure 6. The ontology language is an extension of RDFS (W3C, 2004; Allemang & Hendler, 2011; Obrst, Chase, & Markeloff, 2012) with additional features for learning and evidence representation (Tecuci et al., 2016a).

The learned rules are general IF-THEN structures expressed using the concepts from the ontology. They are learned from explained examples provided by the expert through the employment of the Disciple-EBR multistrategy learning approach, which integrates learning from examples, learning from explanations, and learning by analogy and experimentation, in a mixed-initiative interaction with the expert. Successive versions of this approach are presented in Tecuci (1998) and Tecuci et al. (2002, 2005, 2008, 2016a).

To illustrate the learning process, consider the analysis structure from the top of Figure 3, without the **Collect evidence** requests and the evidence item E1. First, the agent interacts with the ISR expert to determine the important features of the instances from the argument fragment, that is those features that are required for the argument's correctness with respect to the agent's ontology. They include the features that link the instances appearing only in the sub-hypotheses (e.g., "AIS tracking signal1") with the instances appearing in the top-level hypothesis (i.e., "AIS equipment1," "Time1," and "Time2"), as shown in the bottom left of Figure 6. They also include the features that connect the instances from the top hypothesis, representing its meaning with respect to the ontology, such as "Ship1 has as crew Crew1."

In the current implementation, the agent displays a list with all the instances from the argumentation fragment, the expert selects instances from this list, and the agent guides the expert in browsing the ontology fragments connecting those instances, and selecting the important ones. For the example in Figure 3, the selected ontology fragment is shown at the bottom of Figure 6 inside the dotted rectangle: The AIS tracking signal1 of Ship1 with Crew1 was lost between Time1

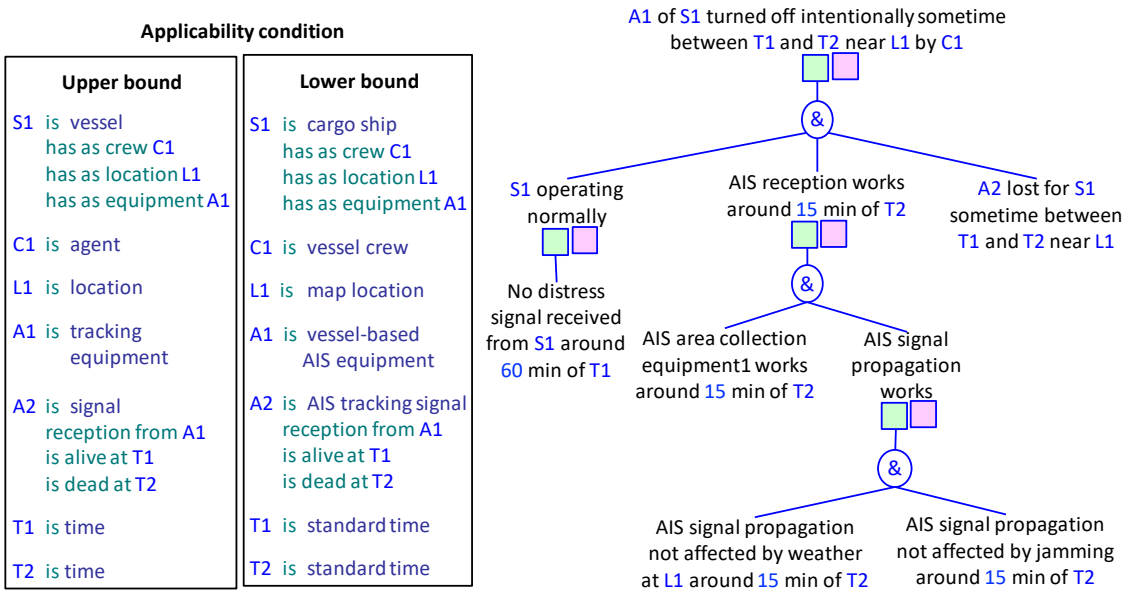

Figure 7. Hypothesis analysis rule learned from the example in Figure 3.

and Time2, when the ship was at Location1. These ontology fragments justify the correctness of the argumentation fragment in Figure 3, called the explanation of this reasoning example.

Then the agent generates the rule from Figure 7. The tree pattern from the right-hand side is obtained by replacing each instance from the reasoning example (e.g., "Crew1") with a variable (i.e., C1). The upper bound of the applicability condition from the left-hand side of Figure 7 is the maximal generalization of the instances (and their features) from the example and the explanation, in the context of the agent's ontology illustrated in Figure 6. The lower bound of the applicability condition is the corresponding minimal generalization.

The possible generalizations of an instance are their super-concepts from the ontology. These generalizations are restricted by the definitions of the features that are related to that instance, that is, by the domains and the ranges of these features. The domain of a feature, such as "reception from," is the set of instances that may have that feature (i.e., "signal"). The range of this feature is the set of its possible values (i.e., "equipment"). Consider, for example, the "AIS equipment1" instance that was replaced with A1 in the rule. As shown in Figure 6, this instance is the value of two features, "reception from" and "has as equipment." Therefore, any generalization of this instance should be in the ranges of these features. As a result, the maximum generalization of "AIS equipment1" is "tracking equipment," obtained as the intersection between the maximum generalization of this instance in the ontology (i.e., "object," the top concept of the ontology, not shown in Figure 6), the range of the feature "has as equipment" (i.e., "equipment," a super-concept of "tracking equipment," also not shown in Figure 6), and the range of the feature "reception from" (i.e., "tracking equipment"). Similarly, the minimal generalization of "AIS equipment1" is "vessel-based AIS equipment," obtained as the intersection between the minimum generalization of this instance in the ontology (i.e., "vessel-based AIS equipment"), the range of the feature "has as equipment", and the range of the feature "reception from."

The learning agent uses the partially learned rules in reasoning to generate new reasoning fragments in future situations. The expert may accept some reasoning fragments as correct, and the

agent will use them as new positive examples of the rule. Those that are rejected represent negative examples. These examples and their explanations are used to refine the rule. In particular, positive examples lead to the generalization of the lower bound condition until it covers them. A negative example may either lead to the specialization of the upper bound condition until it no longer covers it, or to the learning of an *except when* condition, also with an upper bound and a lower bound. The *except when* condition should not be satisfied for the rule to be applicable. In time, the lower bound and the upper bound conditions converge toward one another and, possibly, to an exact applicability condition. The goal is to improve the applicability condition of the pattern so that it only generates correct reasoning fragments. Through a natural expert-agent interaction, this mixed-initiative learning method will lead to learning rules with complex applicability conditions to accurately represent the knowledge of the expert. Notice that, although Disciple-EBR (Tecuci et al., 2016a) implements these rule refinement operators, they are not yet implemented in CAPIP and CAAPT.

Other types of rules (i.e., alert rules, abductive rules, and collection rules) and patterns (i.e., context-independent hypotheses patterns and collection tasks patterns) are learned in a similar way. In particular, from the reasoning structure in Figure 3, the agent learned the hypothesis analysis rule in Figure 7, five context-independent hypotheses patterns (corresponding to the top and the first four of the leaves of the pattern in Figure 7), four collection tasks (corresponding to the specific collection requests in Figure 3), and four collection rules (each reducing a context-independent hypothesis to a collection task).

3.2 Autonomous Multi-Agent Reasoner

The result of teaching the agent is the Reference Knowledge Base (KB), shown at the top of Figure 5, that enables the autonomous Multi-agent Reasoner to perform EBR. The agents of the Multi-Agent Reasoner are copies of the corresponding EBR modules of the Learning and Reasoning Assistant except that they are configured to run autonomously. Notice that the architecture may include multiple copies of an agent (e.g., the Hypothesis Analysis Agent) to speed up the overall analysis process.

The Multi-Agent Reasoner is connected to the application environment through the *Surveillance Manager* and the *Collection Manager*. The *Surveillance Manager* receives alerts from a variety of domain-specific surveillance agents. It then creates uniform representations of such alerts and sends them to the Alert Agent. For each received alert, the *Alert Agent* creates a situation KB containing the ontological representation of the alert, and places these KBs into a queue for the Hypothesis Generation Agent, called the *Hypothesis Generation KB Queue* (see Figure 5). The *Hypothesis Generation Agent* uses the learned abductive rules from the Reference KB to generate, in each KB, the competing hypotheses that may explain the corresponding alert (such as those in Figure 2), and places the updated KBs into the *Hypothesis Analysis KB Queue* for the Hypothesis Analysis Agents.

Each of these *Analysis Agents* uses the learned analysis rules to determine the probabilities of the competing hypotheses through an iterative process of hypothesis decomposition and evidence-based assessment synthesis that was partially illustrated in Figure 3 and Figure 4. Initially, the Analysis Agent decomposes the *competing hypotheses* from a KB into simpler subhypotheses, defines evidence collection requests for the simplest sub-hypotheses, and places the KB into the *Evidence Collection KB Queue* for the Evidence Agent. The *Evidence Agent* translates the evidence collection requests (e.g., the **Collect evidence** requests in Figure 3) into standard representations and sends them to the *Collection Manager* that further translates them into specific API calls to *Collection Agents*. The results returned by the Collection Agents are translated by the Collection Manager and sent to the Evidence Agent to be ontologically represented into the corresponding KB that is re-placed into the *Hypothesis Analysis KB Queue*. The Analysis Agent then attempts to

further extend the analysis. This *Analysis–Collection–Analysis* loop is repeated until there are no new evidence collection requests. At this point the Analysis Agent assesses the probabilities of the competing hypotheses and the KB is placed into a queue for potential user review.

The user supervises this process through the *User Notification and Review Service* that generates an alert when a KB is ready for review. This service facilitates an *"on the loop"* supervision of the autonomous agents by providing real-time situation awareness to the user: which are the competing hypotheses being currently investigated and what is the status of the corresponding analyses; what are each of the agents currently working on, and which analyses are waiting for processing in each of the queues; which evidence collection responses were the most recently received from collection agents, and which evidence collection requests are currently being worked on by collection agents; and finally, which analyses were completed and are waiting for the user's review. The user is notified not only when a hypothesis is confirmed, but also when a conclusion could not be reached due to an incomplete analysis. The user will be able to inspect the incomplete analysis to determine whether that was due to unavailable evidence or due to the encountering of a hypothesis for which the agents require additional training.

All the situation knowledge bases are created by the autonomous agents under the Reference Knowledge Base to inherit the ontology and the learned rules it contains, and they are managed in turn by the *Repository Management Service*. The next section presents the use of this shell in the cybersecurity domain.

4. CAAPT: Cognitive Agent for Advanced Persistent Threats

Modern cyber defense is done in a *cybersecurity operations center* where analysts monitor alerts and log data from available information sources, each having differing levels of credibility, and use them to make a determination about the presence or absence of intrusion activity (Zimmerman, 2014). However, the large and increasing number of alerts and the time required for their manual analysis creates a very complex, expensive, and non-sustainable security environment for network defense organizations which are faced with a shortage of cybersecurity analysts and an average analyst cost that keeps going up. Among the biggest challenges faced by cybersecurity operations centers are those from the *Advanced Persistent Threats* (APTs). These are computer network exploitation groups (many of them state sponsored) that leverage superior resources, knowledge, and tactics to gain and maintain access to targeted networks, and adapt to defenders' efforts to resist them.

4.1 Overview of CAAPT

In an attempt to addressing these problems, we have developed CAAPT. Its architecture is shown in Figure 5. The Learning and Reasoning Assistant is directly instructed by a cybersecurity expert how to investigate cybersecurity alerts. The Reference KB resulting from this training enables the Multi-agent Reasoner to investigate alerts autonomously, as the cyber expert would, but in transparent manner that allows a natural and easy "on the loop" supervision by a cyber operator (bottom right of Figure 5). At the same time, CAAPT can also operate interactively, with "user in the loop", as a trusted collaborator of the human analyst (upper left of Figure 5). The CAAPT's environment is a cybersecurity operations center. The surveillance agents consist of a variety of detection systems, including network-based and host-based intrusion detection systems, anti-virus software, endpoint detection and response tools, or complex detection rules built into security information and event management systems.

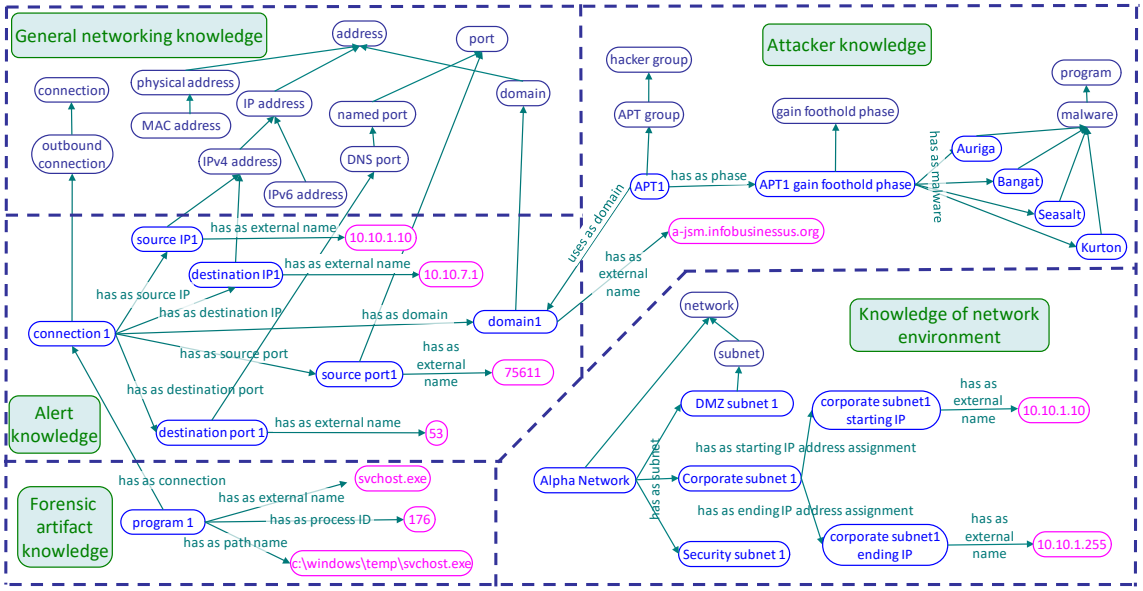

Figure 8. Fragment of the cybersecurity ontology.

As the cyber attack group in our study we selected APT1, the name given by Mandiant (2013) to a group of APT actors, attributed to China's People's Liberation Army unit 61398, who led a years-long campaign of cyber espionage dating back to at least 2004. We chose APT1 because of the abundance of freely available information about it. Reports show an evolution of malware used by them over the eight years they were known to operate. The malware evolution chain used in our research started with the Auriga implant, and then evolved over time to Bangat, Seasalt, and Kurton. Available threat intelligence provides very detailed information on the files, unique strings, Registry keys, persistence mechanisms, and network indicators created during an attack. Using this information, we manually developed attack scenarios recreating the patterns of indicators originally created by each malware program. Attack models were then tested in isolated virtual machines by infecting hosts with APT1 malware and running CAAPT against them.

As indicated, CAAPT is a customization to the cybersecurity domain of the presented approach to autonomous evidence-based reasoning. For example, Figure 8 shows fragments of the various types of knowledge represented in the ontology. *General networking knowledge* represents network devices and protocols, and how they relate to each other in a modern networking environment. *Alert knowledge* represents what specific information is learned when a security alert is raised by a cybersecurity operations center's security infrastructure. Knowledge of the *network environment* covers information specific to the network the cybersecurity operations center is charged with monitoring. *Attacker knowledge* is based on either publicly available threat intelligence or is the result of manual analysis of a threat. *Forensic artifact knowledge* includes knowledge of the data used or left behind by the malware used by an attacker group and where to look for it on a network.

Figure 9 illustrates CAAPT's hypotheses generation process. At the bottom is the alert received by the Surveillance Manager from a surveillance agent, in this case the BRO intrusion detection system (Paxson, 1999). The Surveillance Manager creates a uniform representation in JSON of this

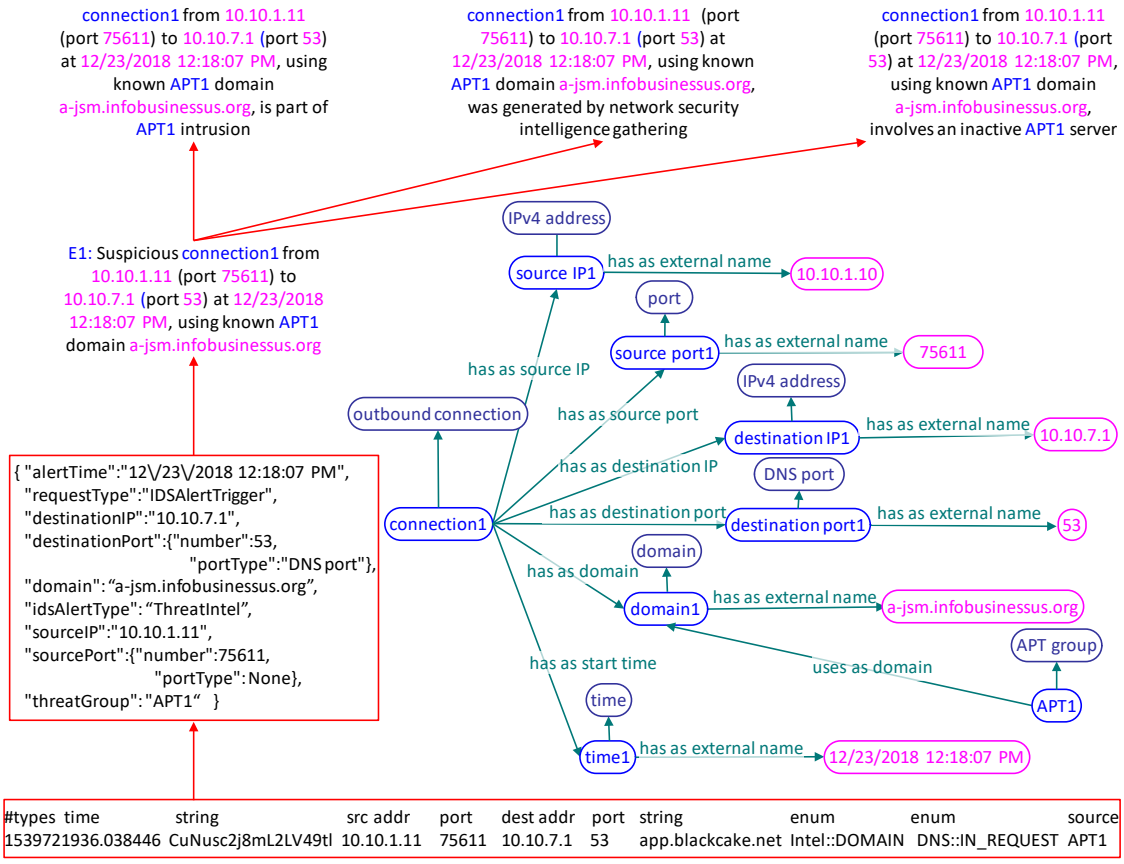

Figure 9. Cybersecurity hypothesis generation.

alert and sends it to the Alert Agent. From this alert, the Alert Agent creates a situation KB containing both the evidential representation of the alert (i.e., "E1: Suspicious connection1 from 10.10.1.11 …") and the ontological representation of the alert (shown in the middle right of Figure 9), and places this KB into the Hypothesis generation KB queue (see Figure 5).

The Hypothesis Generation Agent uses the learned abductive rules from the Reference KB to generate the competing hypotheses from the top of Figure 9 that may explain the alert, and places the updated KB into the Hypothesis analysis KB queue. Notice that there are three generated hypotheses: an intrusion hypothesis (connection1 is part of an APT1 intrusion) and two false positive hypotheses (connection1 was generated by network security intelligence gathering; connection1 involves an inactive APT1 server). An Analysis Agent uses the learned analysis rules to determine the probabilities of the competing hypotheses through an iterative process of hypothesis decomposition and evidence-based assessment synthesis that Figure 10 illustrates.

APT detection requires reasoning over a large set of weak indicators of compromise (IOCs), such as unique strings, filenames, and Registry keys, that must be aggregated to infer the presence of an intrusion. For example, there are two main indicators of the hypothesis from the top of Figure 10: connection1 involves an APT1 command and control server (APT1 C2 server), and the program that made connection1 is an APT1 malware (APT1 malware). The problem is that, in a given situation, the agent may find evidence for the presence of only the first indicator, only the second one, or the presence of both. The * operator corresponds to the disjunction of these three

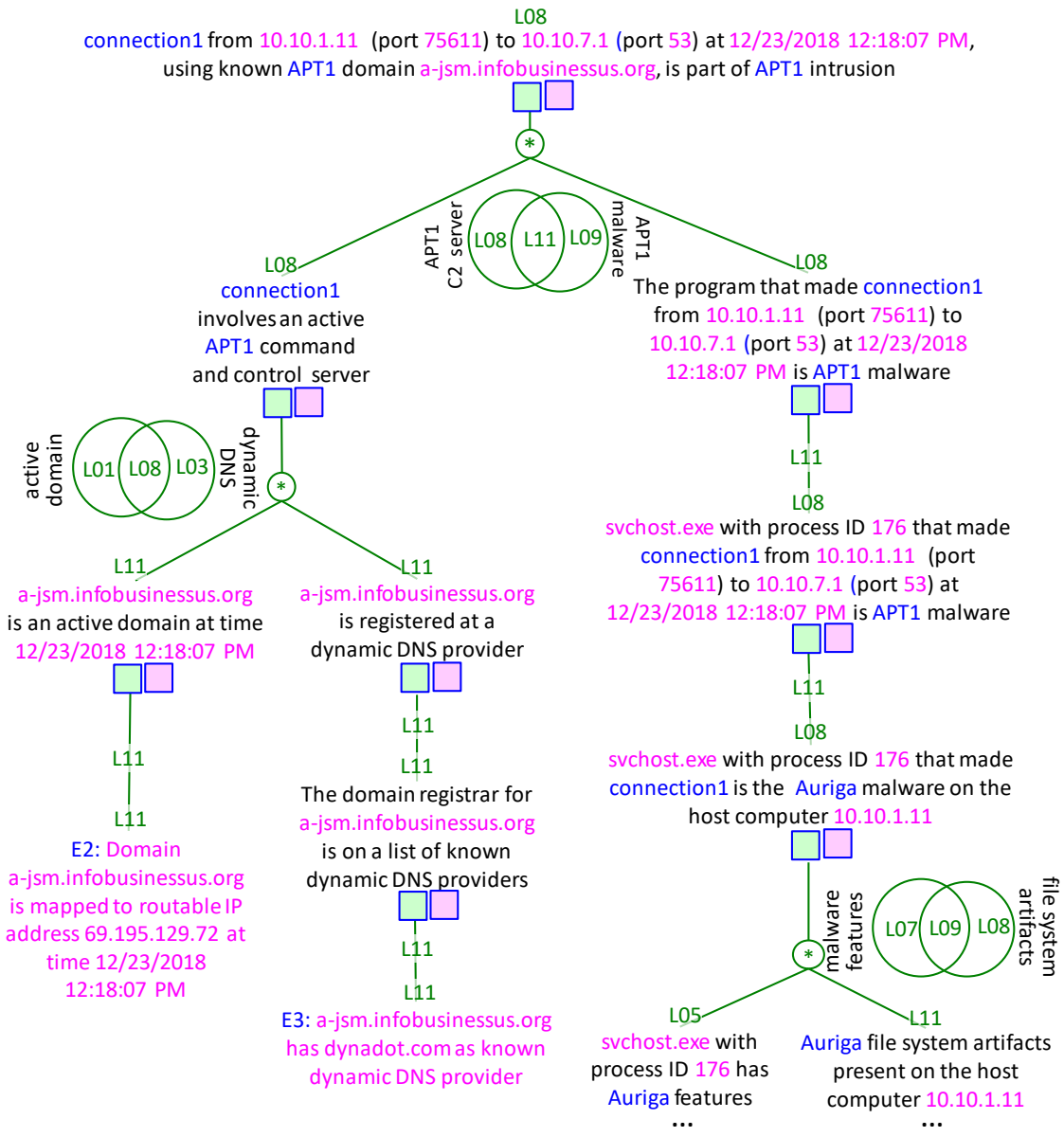

Figure 10. Evidence-based hypothesis analysis.

possibilities. The probabilities inside the two overlapping circles from the top of Figure 10 show the relevance of these three cases to the truthfulness of the top hypothesis:

- The relevance of the "APT1 C2 server" indicator alone is L08 (85-90%) very likely.
- The relevance of the "APT1 malware" indicator alone is L09 (90-95%) very likely+.
- The relevance of these two indicators together is L11 (100%) certain.

We should note that, in the current version of CAAPT, these probabilities are elicited from the cybersecurity expert and represented in the learned rules.

In this example, both indicators have been detected with probability L08 (85-90%) very likely. Therefore their combined relevance to the top hypothesis is L11 (100%) certain. The probability of the top hypothesis is assessed as L08 (85-90%) very likely, the minimum between the probabilities of the two indicators and their combined relevance. The left branch in Figure 10 shows the analysis of the "APT1 C2 server" indicator. First it is decomposed into two sub-indicators: (1) "a-jsm.infobusinessus.org is an active domain at time 12/23/2018 12:18:07 PM" and (2) "a-jsm.infobusinessus.org is registered at a dynamic DNS provider" that is further reduced to "The domain registrar for a-jsm.infobusinessus.org is on a list of known dynamic DNS providers." The items of evidence E2 and E3, each with credibility and relevance L11 (100%) certain, are found for these indicators, and therefore their probabilities are assessed as certain. Their combined relevance to the "APT1 C2 server" indicator is L08 and thus the probability of this indicator is L08.

4.2 Evaluation of CAAPT

Evaluating a system like CAAPT or CAPIP is challenging due to a lack of standardized data for use in comparing it against other systems or approaches. It is also challenging due to a lack of systems similar to them. It is a novel approach, both with respect to autonomous evidence-based reasoning in general and with respect to APT detection in particular. As such, the only reasonable approach to compare CAAPT would be to manual analysis by an expert, but even this is problematic because of lack of data on manual analysis.

We designed and performed experiments to test both the training of CAAPT and its ability to detect configuration changes in the same malware and new malware versions as the attackers' tool set evolved over time. The experiment simulated a subset of the historical evolution of APT1 malware: Auriga → Auriga variants → Bangat → Bangat variants → Seasalt → Seasalt variants → Kurton → Kurton variants. This enabled us to test five claims:

- *Ability to automatically detect the training malware:* Once trained with a malware, CAAPT can automatically detect it. This creates a baseline for the evaluation.
- *Ability to detect variants of the training malware:* Once trained with a malware (e.g., Auriga) CAAPT can automatically detect new variants of it (i.e., Auriga variants).
- *Some ability to detect evolved malware:* Once trained with some members of a malware family (e.g., Auriga and Bangat of APT1), it may be able to detect an intrusion by a new member of the (APT1) family.
- *Limited incremental training needed to detect a new malware from the same family* (e.g., limited incremental training needed to detect Bangat, after it was trained to detect Auriga).
- *Efficient and high quality analysis:* CAAPT can rapidly detect APT1 intrusions through a rigorous and transparent analysis, as judged by the training expert.

We started the evaluation experiment with developing a cyber ontology (see Figure 8) and with training CAAPT to analyze the Auriga malware of APT1 (see Figure 10), based on the expertise of one of us (Steven Meckl). After that we tested CAAPT's detection capabilities in three scenarios:

(a) With the Auriga intrusion used in training (to create a baseline for the evaluation).
(b) With an intrusion by a variant of Auriga. This variant used a different APT1 domain to trigger the security alert, and the malware process %SYSTEMROOT%\Temp\svchost.exe did not contain unique APT1 strings.
(c) With a Bangat intrusion. Bangat does not have the library files riodrv32.sys and netui.dll, uses a different regular expression for its temporary file names, stores its data files in different folders, and uses different Windows Service names for its persistence mechanisms.

Table 2. Summary of the CAAPT experimental results.

1.Intrusion Detection	***Intrusion by the Auriga used in training***	***Intrusion by variant of Auriga used in training***	***Intrusion by Bangat***
Auriga	L08 (85-90%) very likely	L08 (85-90%) very likely	L06 (75-80%)
APT1	L08 (85-90%) very likely	L08 (85-90%) very likely	L08 (85-90%) very likely
Duration	143 seconds	121 seconds	119 seconds
2.Intrusion Detection	***Intrusion by the Bangat used in training***	***Intrusion by variant of Bangat used in training***	***Intrusion by Seasalt***
Auriga	L06 (75-80%)	L06 (75-80%)	L01 (50-55%) barely likely
Bangat	L08 (85-90%) very likely	L08 (85-90%) very likely	L01 (50-55%) barely likely
APT1	L08 (85-90%) very likely	L08 (85-90%) very likely	L08 (85-90%) very likely
Duration	265 seconds	228 seconds	274 seconds
3.Intrusion Detection	***Intrusion by the Seasalt used in training***	***Intrusion by variant of Seasalt used in training***	***Intrusion by Kurton***
Auriga	L01 (50-55%) barely likely	L01 (50-55%) barely likely	L01 (50-55%) barely likely
Bangat	L01 (50-55%) barely likely	L01 (50-55%) barely likely	L03 (60-65%) likely
Seasalt	L08 (85-90%) very likely	L08 (85-90%) very likely	L00 (0-50%) lacking support
APT1	L08 (85-90%) very likely	L08 (85-90%) very likely	L08 (85-90%) very likely
Duration	382 seconds	406 seconds	344 seconds
4.Intrusion Detection	***Intrusion by the Kurton used in training***	***Intrusion by variant of Kurton used in training***	
Auriga	L01 (50-55%) barely likely	L01 (50-55%) barely likely	
Bangat	L03 (60-65%) likely	L03 (60-65%) likely	
Seasalt	L00 (0-50%) lacking support	L00 (0-50%) lacking support	
Kurton	L07 (80-85%)	L07 (80-85%)	
APT1	L08 (85-90%) very likely	L08 (85-90%) very likely	
Duration	587 seconds	631 seconds	

In the second phase of the experiment we have extended the ontology and trained CAAPT to detect the Bangat malware used in the testing of the first phase. This process was repeated with Seasalt (third phase) and Kurton fourth phase). The main results are summarized in Table 2, phase by phase, and discussed in the following with respect to the above five claims.

The first column of Table 2 lists the malware detected by CAAPT in each phase of the experiment, when the intrusion was made by the malware shown in the leading row of the other columns. Consider, for example, the second row in Table 2. CAAPT detected an Auriga intrusion with probability L08 when the intrusion was made by the Auriga used in training, with the same probability of L08 when the intrusion was made by the variant of the Auriga used in training, and with probability of L06 when the intrusion was made by Bangat.

Ability to automatically detect the training malware. As shown in column 2 of the table, CAAPT detected the intrusion with the malware used in training with probability L08 (85-90%) for Auriga, Bangat and Seasalt, and with probability L07 (80-85%) for Kurton.

Ability to detect variants of the training malware. As shown in column 3 of the table, CAAPT detected the intrusion with a variant of the malware used in training with the probability of L08 for the Auriga, Bangat, and Seasalt variants, and with probability of L07 for the Kurton variant.

Table 3. The evolution of the ontology during agent training.

	Initial KB	+ Auriga	+ Bangat	+ Seasalt	+ Kurton	Total
Concepts	72	48	0	0	0	120
Instances	76	27	3	4	0	110
Features	17	13	10	10	7	57
Facts	32	47	40	40	32	191
Total	197	135	53	54	39	478

Some ability to detect evolved malware. As shown in column 4, after being trained to detect Auriga and invoked to analyze an intrusion using Bangat, CAAPT still reported an APT1 intrusion with probability L08 (85-90%), but the probability of being Auriga was lower L06 (75-80%). In the case of analyzing Seasalt after being trained on Auriga and Bangat, CAAPT still detected an APT1 intrusion with probability L08 (85-90%), but the probability of being Auriga or Bangat was only L01 (50-55%). A similar result was obtained in the case of analyzing Kurton: After being trained on Auriga, Bangat, and Seasalt, CAAPT still detected an APT1 intrusion with probability L08 (85-90%), but the probability of being Auriga was L01 (50-55%), of being Bangat was L03 (60-65%), and of being Seasalt was L00 (0-50%).

Several remarks about the estimated probabilities are appropriate. Notice that, in the first phase of the experiment (first four rows in Table 2), the sum of the probabilities of "Auriga intrusion" and "APT1 intrusion" is over 100%. This is because these two hypotheses are not disjoint. Indeed, "APT1 intrusion" means any intrusion performed by the APT1 attacker group, using any of their malware tools, including Auriga. Notice also that, as expected, in each of the four phases of the experiment, the probability of "APT1 intrusion" is greater than or equal to the probability of the intrusion with any of the considered members of the APT1 family (i.e., Auriga, Bangat, Seasalt, and Kurton).

Let us now consider the results of the second phase of the experiment, shown in rows 5 to 9 of Table 2. Notice in column 2 that the probability of "Auriga intrusion" is L06 (75-80%) and the probability of "Bangat intrusion" is L08 (85-90%). This is not a contradiction because these two hypotheses are not disjoint. The Bangat malware is an evolution of the Auriga malware and therefore it has many features in common with Auriga. When checking for an intrusion with Auriga, the system looks for the presence of the features of the Auriga malware on the infected computer, but some of these features are also the features of Bangat, so it is possible that the computer is infected by both Auriga and Bangat. Therefore, Auriga intrusion with probability L06 (75-80%) covers the case where the Auriga intrusion is accompanied by a Bangat intrusion. Similarly, Bangat intrusion with probability L08 (85-90%) is based on the detected Bangat features on the host computer which also includes some Auriga features. Thus, this probability also covers the case when there is both a Bangat an Auriga and intrusion.

Limited incremental training needed to detect a new malware from the same family. Table 3 shows the evolution of the ontology during agent training. Column 2 shows the number of domain-independent ontological elements present in the initial knowledge base. To model the detection of the Auriga malware, the ontology was extended with 48 concepts, 27 instances, 13 feature definitions, and 32 facts, for a total of 135 new elements. However, modeling the Bangat malware required only 53 additional elements (no new concepts, three new instances, ten new feature definitions, and 40 new facts). Similarly, modelling Seasalt and Kurton required the extension of

Table 4. The evolution of the rules during agent training.

	Auriga	+ Bangat	+ Seasalt	+ Kurton	Total
Hypothesis patterns	28	1	10	1	40
Alert rules	2	0	0	0	2
Abductive rules	2	0	0	0	2
Hypothesis analysis rules	13	1	7	2	23
Collection tasks	15	1	5	0	21
Collection rules	15	1	5	0	21
Collection agents	8	0	1	0	9
Total elements	83	4	28	3	118

the ontology with only 54 and 39 new elements, respectively. After training, the final ontology contained only 478 elements, with over 40% of them being part of the initial knowledge base.

Table 4 shows the number of new knowledge elements learned during each phase of the experiment. As a result of the initial training of CAAPT to detect Auriga, it learned 28 context-independent hypotheses patterns, two alert rules, two abductive rules, 13 hypotheses analysis rules, 15 collection tasks, and 15 collection rules, for a total of 75 learned elements. Eight collection agents were also defined. However, further training of CAAPT to detect Bangat resulted in only one new context-independent hypothesis pattern, one new hypotheses analysis rule, one new collection task, and one new collection rule. Thus, a total of only four new elements needed to be learned to detect Bangat. For Seasalt, which is a more significant evolution of Bangat, a total of 27 new elements needed to be learned and one new collection agent defined. Finally, for Kurton however, only three new elements needed to be learned.

Overall, CAAPT had to learn only 40 context-independent hypotheses patterns, two alert rules, two abductive rules, 23 hypotheses analysis rules, 21 collection tasks, and 21 collection rules, in order to detect intrusions from the four families of the APT1 malware discussed above: Auriga, Bangat, Seasalt, and Kurton (nine collection agents were also defined).

Efficient and high quality analysis. The Duration rows in Table 2 provide the total run times to detect an intrusion. This time increased from around two minutes, when CAAPT was checking for Auriga intrusions only, to around ten minutes when CAAPT was checking for Auriga, Bangat, Seasalt, and Kurton intrusions. However, most of this time was spent by waiting for the Collection Manager to return the results requested from the collection agents. The actual run time for the development and evaluation of the reasoning trees (see Figure 10) only increased from around two seconds, when CAAPT was checking for Auriga intrusions only, to around six seconds when CAAPT was checking for Auriga, Bangat, Seasalt, and Kurton intrusions. Additionally, the training expert judged the generated reasoning trees as correct, rigorous and very clear.

5. Conclusions

We have presented a general approach to the development of instructable cognitive agents for automated evidence-based reasoning, a process modelled as continuous discovery of evidence, hypotheses, and arguments. This approach was implemented in a learning agent shell for evidence-based reasoning consisting of a mixed-initiative learning and reasoning assistant, and an autonomous multi-agent reasoner. A subject matter expert teaches the learning and reasoning assistant by demonstrating and explaining each reasoning step involved in the investigation of a

specific alert. As a result, the assistant learns rules for generating hypotheses that explain alerts, rules for discovering relevant evidence, and rules for testing the hypotheses. Specialized collaborative agents use the learned rules to automatically conduct alert investigations. The generality of the proposed approach is supported by the development of two autonomous agents, one for ISR and the other for cybersecurity. Experimental results of training and testing the cybersecurity agent show that, after being trained to detect a specific malware, the agent was able to detect variants of that malware, and that it only required limited incremental training to detect other members of the family of that malware.

We plan to research more advanced evidence-based reasoning methods. This includes a more efficient approach to automatic hypothesis generation as a multi-step abductive process where each abductive step involves generating competing hypotheses, collecting evidence, and testing these hypotheses, to significantly prune the hypothesis space. It also includes a more advanced method for hypothesis testing, through a natural and easy to understand integration of logic and multi-probabilistic reasoning views, to more accurately assesses both the probability of each hypothesis and the confidence in this probability. We plan to further develop the rule learning and refinement approach, and improve the methods for learning different types of rules, particularly the alert rules and the evidence collection rules that involve the generation of ontology fragments. We also plan to further develop the agent shell, the ISR agent, and the cybersecurity agent, and to explore other applications of the presented approach, such as personal health monitoring and fraud detection in financial services.

Acknowledgements

Tom Bartee developed the MITRE's Integrated Environment for Persistent Intelligence that was integrated with CAPIP. Chirag Uttamsingh contributed to several related EBR projects of the Learning Agents Center. We are also grateful to the anonymous reviewers for their very helpful comments. The research reported in this paper was performed in the Learning Agents Center and was supported in part by the Air Force Research Laboratory under contract number FA8750-17-C-0002, by MITRE Corporation under research agreement number 114615, by the National Science Foundation under grant number 1611742, and by George Mason University. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government.

References

Allemang, D., & Hendler, J. (2011). *Semantic web for the working ontologist: Effective modeling in RDFS and OWL*. Waltham, MA: Morgan Kaufmann.

Cohen, L. J. (1977). *The probable and the provable.* Oxford, UK: Clarendon Press.

Cohen, L. J. (1989). *An introduction to the philosophy of induction and probability.* Oxford, UK: Clarendon Press.

Eco, U. (1983). Horns, hooves, insteps: Some hypotheses on three types of abduction. In U. Eco, & T. Sebeok (Eds.), *The sign of three: Dupin, Holmes, Peirce*, 198-220. Bloomington, IN: Indiana University Press.

International Maritime Organization. (2019). *Automatic Identification Systems.* Retrieved May 22, 2019, from http://www.imo.org/en/OurWork/Safety/Navigation/Pages/AIS.aspx.

Mandiant (2013). *APT1: Exposing One of China's Cyber Espionage Units.* Retrieved August 24,

2019, from https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf.

Obrst, L., Chase, P., & Markeloff, R. (2012). Developing an ontology of the cyber security domain. *Proceedings of the Seventh International Conference on Semantic Technologies for Intelligence, Defense, and Security* (pp. 49–56). Fairfax, VA: CEUR.

Paxson, V. (1999). Bro: A system for detecting network intruders in real-time. *Computer Networks, 31*, 2435–2463.

Peirce C. S. (1955). Abduction and induction, 1901. In J. Buchler (Ed.), *Philosophical writings of Peirce*, 150-156. New York, NY: Dover Publications.

Schum, D. A. (2001a). *The evidential foundations of probabilistic reasoning*. Evanston, IL: Northwestern University Press.

Schum, D. A. (2001b). Species of abductive reasoning in fact investigation in law. *Cardozo Law Review, 22*, 1645–1681.

Tecuci, G. (1998). *Building intelligent agents: An apprenticeship multistrategy learning theory, methodology, tool and case studies.* San Diego, CA: Academic Press.

Tecuci, G., Boicu, M., Marcu, D., Stanescu, B., Boicu, C., Comello, J., Lopez, A., Donlon, J., & Cleckner W. (2002). Development and deployment of a Disciple agent for center of gravity analysis. *Proceedings of the Eighteenth National Conference of Artificial Intelligence and the Fourteenth Conference on Innovative Applications of Artificial Intelligence* (pp. 853–860). Edmonton, Alberta: AAAI Press.

Tecuci, G., Boicu, M., Boicu, C., Marcu, D., Stanescu, B., & Barbulescu, M. (2005). The Disciple-RKF learning and reasoning agent. *Computational Intelligence, 21*, 462–479.

Tecuci, G., Boicu, M., Marcu, D., Boicu, C., & Barbulescu, M. (2008). Disciple-LTA: Learning, tutoring and analytic assistance. *Journal of Intelligence Community Research and Development*, July. Retrieved September 4, 2019, from http://lac.gmu.edu/publications/2008/Disciple-LTA08.pdf.

Tecuci, G., Marcu, D., Boicu, M., & Schum, D.A. (2016a). *Knowledge engineering: Building cognitive assistants for evidence-based reasoning*. New York, NY: Cambridge University Press.

Tecuci, G., Schum, D. A., Marcu, D., & Boicu, M. (2016b). *Intelligence analysis as discovery of evidence, hypotheses, and arguments: Connecting the dots*. New York, NY: Cambridge University Press.

Thagard, P. R. (1993). *Computational philosophy of science.* Cambridge, MA: MIT Press.

Wigmore, J. H. (1913). The problem of proof. *Illinois Law Review*, *8*, 77–103.

W3C (2004). *RDF Schema 1.1*. Retrieved July 7, 2019, from http://www.w3.org/TR/rdf-schema/.

Zadeh, L. (1983). The role of fuzzy logic in the management of uncertainty in expert systems. *Fuzzy Sets and Systems, 11*, 199–227.

Zimmerman C. (2014). *Ten strategies of a world-class cybersecurity operations center*. Bedford, MA: The MITRE Corporation.