CORA: A Flexible Hybrid Approach to Building Cognitive Systems

Stephen Lee-Urban STEPHEN.URBAN@GTRI.GATECH.EDU
Ethan Trewhitt ETHAN.TREWHITT@GTRI.GATECH.EDU
Ian Bieder IAN.BIEDER@GTRI.GATECH.EDU
Joel Odom ODOM@GATECH.EDU
Timothy Boone TIMOTHY.BOONE@GTRI.GATECH.EDU
Elizabeth Whitaker ELIZABETH.WHITAKER@GTRI.GATECH.EDU
Georgia Tech Research Institute, 75 Fifth St. NW, Atlanta, GA 30332 USA

Abstract

The Cognitive Reasoning and Representation Architecture (CORA) is a flexible hybrid approach and toolset that enables human-like reasoning across a variety of problem domains. It combines multiple existing reasoning techniques and implementations along with a framework to facilitate sharing of knowledge among reasoning modules, including (but not limited to) case-based reasoning (CBR), Bayesian networks, and fuzzy reasoning. CORA enables the exchange of various knowledge types across reasoning modules via point-to-point, point-to-multipoint, and shared-memory communications mechanisms. Three problem domains are demonstrated: cognitive wireless communications, electronic sensing for novel radar waveforms, and intelligence surveillance and reconnaissance (ISR) sensor retasking. CORA's multi-platform implementation supports a variety of software languages to enable reasoning processes to take place using the best tools for each reasoning process, including the integration of several well-known open-source reasoning tools.

1. Introduction

The objective of Georgia Tech Research Institute's (GTRI's) Cognitive Reasoning and Representation Architecture (CORA) project is to explore hybrid cognitive reasoning and representation approaches with a primary focus on the spectrum reasoning and electronic warfare (EW) domains (and unrelated domains with a set of similar characteristics), and to prototype a framework that will enable the support of multiple scenarios in these domains. The framework will enable flexible configuration of case-based reasoning, Bayesian networks, fuzzy reasoning approaches and reasoning under uncertainty to support spectrum and EW problems. The exploration of these reasoning techniques includes learning and meta-reasoning to enable improvement of reasoning over time. This framework will address the need of these domains to respond and adapt to unexpected inputs and to include contextual information from multiple sources to support decision-making in this domain. We have chosen specific tools and programming languages for the prototypes, but our intent is to enable the use of different implementations tailored to the needs of the problem and allowing for integration with heterogeneous modules. The strength of the architecture is providing representations that translate between the causal models, case-based models and fuzzy models allowing each component to provide a different type of inference to the solution.

GTRI customers require the capability to respond to unexpected situations or threats that have not been previously seen and to adapt their solutions in the spectrum operations and electronic warfare domains to respond in near-real-time to these situations. We have done work on a DARPA project called Behavioral Learning for Electronic Warfare, supplying case-based learning to categorize threats and to supply countermeasure parameters, and we are collaborating with other groups at GTRI to supply cognitive reasoning (starting with Case-based Reasoning) to support their electronic warfare applications. When compared to homogeneous reasoning techniques, hybrid reasoning techniques tend to be more robust and able to respond to a wider range of unexpected threats or situations in more intelligent and adaptive ways. We have chosen case-based reasoning and learning, Bayesian network learning and reasoning, fuzzy reasoning and reasoning about uncertainty as the approaches to explore and develop because they possess a wide variety of problem-solving characteristics that will have a large effect on the ability to respond to unexpected situations in these domains. We are experimenting with and prototyping a configurable framework that allows the cognitive reasoning and representation components to integrate with other spectrum or EW systems and allows different cognitive reasoning paradigms to operate in a selected sequence or opportunistically to solve a given problem type. We are using a cognitive architecture that includes reflexive, deliberative and reactive components.

1.1 Abstract Representation of the Cognitive Components of CORA

Figure 1 shows a model of cognitive architectures. Representations similar to this are widely accepted in the intelligent systems field and have been used by DARPA and other research organizations to describe this technical approach to designing and building cognitive-adaptive systems. The figure shows three layers of reasoning processes: reactive, deliberative and reflective.

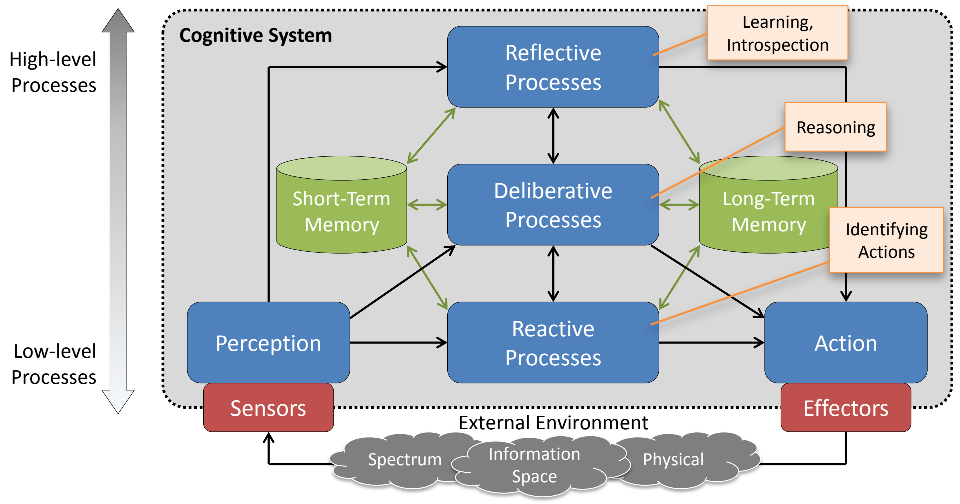

Figure 1. Abstract model of a horizontally layered cognitive system.

Reactive processes are simple, quick decisions that once learned are automatically performed. Examples of reactive responses are the basic moves of a robot to avoid collisions or falls, or a wireless system retransmitting a message that was lost due to a collision with another transmission.

Deliberative processes require more knowledge and more processing. This is often the central piece of a cognitive system. Deliberative processes are used to make complex decisions that may require complex algorithms, sets of rules or cases, more intelligence and knowledge. These include reasoning about missions, the handling of constraints, business rules, rules of engagement or resource management decisions. Responding to unexpected situations or dynamically adapting responses to new information requires deliberative reasoning.

Reflective processes are metacognition components that reason about the decisions or solutions provided by the system. These processes attempt to identify the strengths and weakness of the solutions provided by the system. They are used to guide learning or self-improvement, that is, the updating of the deliberative or reactive reasoning processes or the long term memory so that the cognitive system performs better on future problems. Adaptive EW requires a system that can learn and improve its performance. It should work toward being able to capture, in real-time, the classification, identification, and countermeasure techniques and evaluate the success and accuracy of those results, storing successful solutions and updating approaches so that the most successful approaches and lessons learned from them are reused.

Short-term memory in a cognitive systems is information about the current situation and current problem being solved. Examples include communications currently taking place, receivers, online emitters, external sensor feeds ISR assets, or situational awareness. These all describe the system's memory of recent events.

Long-term memory in a cognitive system is the domain and contextual knowledge that the system uses to reason. This includes human-like knowledge available to and used by a domain expert when making decisions. This long-term memory includes knowledge about the spectrum and how it works or how it is typically used, knowledge of the context or the environment in which a system exists, geographic maps, military mission descriptions and priorities, societal communication norms, human behavior, and seasonal and calendar activities. Long term memory will also include any ontologies that the system uses to reason with other system components about shared concepts and vocabulary.

1.2 Related Work

In *Intelligent Behavior in Humans and Machines* (2012), Langley presents a characterization of the cognitive systems paradigm that, in addition to highlighting the importance of connections to psychology and human cognition, also notes the tendency for the paradigm to incorporate high-level processing, structured representations, and heuristic methods. Further, he identifies a subfield of cognitive systems work that attempts to develop a unified theory of the human mind, typically through memory processes involving the careful construction and use of multiple types of memory, reasoning processes that use those memories, and learning processes that effect the memory and reasoning processes.

Our work on CORA is motivated by the desire to achieve autonomous reactive and proactive human-level performance in both familiar and novel situations that unfold in problem areas common to GTRI customers. The solicitation and effective use of knowledge from human experts, compliance with standard operating procedure, and use of contextual information from multiple sources to support decision-making is a common requirement in these problem areas. To do so, we adopt an approach that brings together logical, case-based, and probabilistic approaches in a configurable framework.

This approach is driven by our experience that has shown hybrid reasoning techniques to be more robust and able to respond to a wider range of unexpected threats or situations in more intelligent and adaptive ways, rather than being driven by a desire to attempt to create a unified theory of the mind. We therefore do not adopt a unified theory approach (e.g. symbolic), make no claim about what human cognition is nor how it can be implemented in a general framework computationally (c.f. the ACT-R architecture in Anderson et al., 2004; the SOAR architecture in Laird, 2012; the ICARUS architecture in Choi, 2010), nor does our framework use a specialized language for its control. We do, nevertheless, rely heavily on well-understood artificial intelligence (AI) techniques.

In CORA we make use of a hybrid approach that incorporates both high and low level processing, a variety of structured representations, and domain-specific heuristic methods. In this respect, CORA is more similar to the approaches presented by GILA (Zhang et al. 2009). Our team supplied case-based learning and reasoning as part of the GILA project for the DARPA Integrated Learning Project. CORA also shares similarities with Model Docking as in (Trewhitt, Whitaker, Briscoe, & Weiss, 2011), TouringMachines (Fergurson, 1992), and Polyscheme (Cassimatis, 2001). In CORA, in order to manage the complexity of multiple interacting heterogeneous expert systems, we adopt a variety of the design principles, insights and techniques espoused by the multi-agent systems community (for an introduction, see Wooldridge, 2009).

2. Learning, Inference, and Problem Solving in CORA

There are many approaches to addressing cognitive adaptive reasoning in the spectrum reasoning, electronic warfare (EW) and related areas. There are a few AI techniques of particular interest because they provide a portfolio of reasoning approaches that address different aspects of these problems for a number of use cases in this domain. For hybrid reasoning, each approach can provide support for the others, allowing for flexible configuration moving toward collaborative, distributed multi-agent problem solving. This section contains a brief description of each of the approaches that we employ in CORA, starting with case-based reasoning, which plays a central role.

2.1 Case-Based Reasoning

Case-based reasoning (CBR) consists of solving new problems by reasoning about past experiences (Kolodner, 1993). An advantage of case-based reasoning over other machine learning approaches is that it can be successfully applied to problems that are ill-defined and for which the number of training examples may be insufficient for other types of reasoning. CBR allows us to deal with minimal training examples, missing data, ill-defined feature spaces and uncertainty. CBR relies on the assumption that if two problems are “close” in the problem space then their solutions are likely to be “close” in the solution space. Experiences are captured and stored as a collection of *cases* stored in a *case library*. Each of these cases contains a past problem and the associated solution and is indexed by a set of *features* that characterize the case.

Solving new problems involves (a) structuring the new problem into a format compatible with the case structure by a *Feature Extractor*, which takes the information from the problem state and formats them into a *target case*, (b) *retrieving* one or more relevant cases from the case library using a *similarity metric*, (c) *reusing* or *adapting* one or more of the solutions contained in the retrieved cases, using specialized adaptation knowledge, to form a new solution applicable to the target case which may then be used to solve the current problem instance, and (d) optionally *retaining* the target case along with the new solution as new knowledge in the case library. *Offline learning* involves cases that are trained explicitly (sometimes using a specialized case learner) prior to performance and have a specific solution associated with known training data from an existing problem, human expert trace, or simulation results. *Online learning* refers to the process of creating and retaining new cases in the case library during performance, whenever attempts to classify an incoming set of threat features yield retrieved cases that are sufficiently dissimilar to the target case. Cases learned through online learning can then be retrieved during future classification efforts, increasing the likelihood that similar new cases will be recognized without being explicitly trained. These cases can later be updated with a more well-defined solution.

A similarity metric is based on the distance between a case in the case library and the target case. This distance is an abstract distance in the feature space, not a geographical distance. A weighted Euclidean feature distance is a common approach. In general, $similarity_i = 1 - normalized\ distance_i$. The simplest approach is to compare the target case with every case in the case library and select one or more “nearest neighbors.” Weights may be recommended by a domain expert, but are often refined through experimentation.

There are some situations in which a Euclidean similarity metric will not provide good results. While many systems employ a normalized Euclidean similarity metric, the CBR module in CORA is extensible and has the ability to use multiple similarity metrics. For example, if the similarity of retrieved cases is frequently insufficient to allow for a definitive match, a taxonomy may be used to provide further knowledge to inform retrieval. Taxonomy-based retrieval uses the associated confidence levels to determine whether it is possible to say with a greater certainty what common ancestor is shared by the best case matches. Testing real-world cases and comparing their outcomes can identify potential variants in feature specifications in order to select the most accurate similarity metric for a particular case library.

The function of the *Adaptation* module in a CBR system is dependent on domain knowledge of the problem space. Its purpose is to modify, if necessary, the old solution to fit the new problem. This is a type of analogical reasoning in which the old and new problems are compared so that the old solution may be translated to create the new solution. Many approaches are used in adaptation modules and often hybrid reasoning techniques are required. We have successfully applied rule-based approaches and problem domain-specific algorithms from domain experts.

The process of *case-based learning* is performed by observing events and their associated solutions, extracting the problem descriptions, features, and solutions, then storing them as cases in a case library. This might require several specialized case libraries, one for each type of problem. For each of those libraries, we propose two learning modules: one capable of learning cases, and one for adaptation knowledge. Adaptation knowledge is expressed as a set of transformation rules and a set of constraints. Adaptation rules capture how to transform the solution from the retrieved case to solve the problem at hand, and the constraints specify the combinations of values that are permitted in the solutions being generated.

2.2 Fuzzy Reasoning

Fuzzy Logic is an AI technique developed to support the modeling of imprecise concepts and modeling with imprecise dependencies among the entities about which a system is reasoning (Zadeh, Yuan, & Klir, 1996). It is a superset of classical logic that introduces the concept of "degree of membership" in a set. For example, the frequency of an amplitude of a radio signal can be described as "high" to a certain degree rather than by a specific power level in decibels. Not simply "high" or "not high", the value is considered to have a "high"ness value between zero and one.

Fuzzy reasoning is often used because it provides an efficient way to represent linguistic and subjective attributes of the real world in a computational representation that can be used to reason in an intelligent system. It is a natural representation for acquiring and representing knowledge of human experts who do not always reason in precise quantitative algorithms. A fuzzy reasoner is particularly useful in multi-parameter decisions and situations, at providing non-linear controls for a system, capturing and representing expert knowledge, modeling human behavior and providing approximate reasoning when precise information is not available.

Creating a fuzzy reasoning system requires the definition of fuzzy variables and fuzzy sets along with a set of fuzzy rules to reason on these variables. A Fuzzy Control Language or FCL is used to define and invoke the fuzzy inference operations. In our hybrid experiments we apply fuzzy reasoning to the adaptation of cases in CBR, and to knowledge acquisition from experts.

2.3 Reasoning with Bayesian Networks

A Bayesian Network (BN) is a directed graph where each node represents a random variable and each directed edge represents probabilistic dependence between nodes. Modelling with BNs is an attempt to capture the complete joint probability distribution of the modeled problem (Pearl, 2014). A BN gives the ability to infer the probability of unobserved events based on evidence known about related events and prior probabilities; this enables diagnostic inference (observe effects, infer causes), causal inference (observe causes, infer effect likelihoods), and targeted information gathering using value of information calculations.

As a simple example, consider the dependence between the day of the week and the travel time between two points in a city (disregarding other variables like time of day). It is natural to think of the day of the week as influencing the driving time; for example Fridays may statistically have the longest driving time. A BN model of the problem would have two nodes, one called "day of the week" and the other "driving time", and a directed edge from the former to the latter.

The arrow from "Day of Week" to "Driving Time" shows that the day of the week influences the time that it takes to drive between the points. Suppose that we are given the driving time for a particular commute and we want to know the likelihood that this particular commute was performed on a Sunday. We write this question mathematically as

$$\Pr[\ d = \text{“Sunday”} \mid t = \tau\ ]$$

where d is the day in question, t is the time variable, and τ is the given time assigned to t in this particular circumstance. The equation asks, "What is the probability that the day, d, is Sunday, conditioned on the knowledge that the time to drive, t, is τ?" For the simple BN above this equation, along with the prior and conditional probabilities, completely describes the problem.

3. Generalized, Integrated Domain-Configurable Architecture of CORA

In this section, we briefly present the software design of CORA's configurable framework that enables the flexible hybrid use of the reasoning approaches described earlier. The focus is primarily on the modular CORA components. This framework allows the CORA components to integrate with other spectrum or EW systems, and allows different cognitive reasoning paradigms to operate in a selected sequence or opportunistically to solve a given problem type. *Figure 2* depicts the conceptual layers that support these requirements. On the left of the figure, we identify the high level elements that are a part of each of CORA's pluggable modules; on the right we highlight example functions associated with those layers.

The *Connectivity* layer is the base of the modular, distributable architecture. It is the set of channels through which modules can communicate. For maximum flexibility we use a free, high-speed, language and platform independent tool that improves on socket based communication by providing a single interface that supports communication within a single process, communication between processes on a single computer, and communication between processes on multiple computers. It has built in support for patterns like request-reply, publish-subscribe, push-pull, and router-dealer, and is data-agnostic, working directly in bytes. We add to this a common logging subsystem to ease tracing the flow of information in the cognitive system (for development, demonstration and debugging). While each module may use its own process-native logging facilities, the common logging subsystem acts as a distributed logging facility with a web frontend.

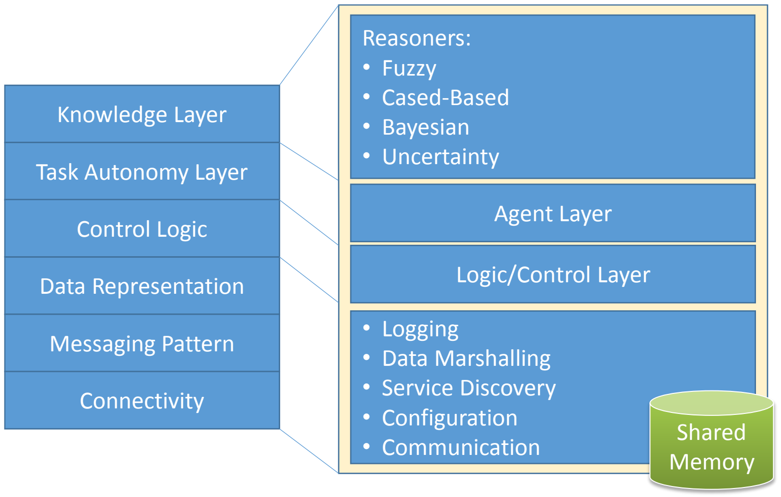

Figure 2. Abstract presentation of the components in CORA's configurable framework.

The *Messaging Pattern* layer provides a common configuration subsystem that captures the way in which individual CORA modules interact with one-another (often problem domain dependent). It is built on top of the connectivity layer's communication patterns and extends them by supporting both manually configured endpoints and protocols as well as by providing a simple service-discovery approach. The former is a unified means of statically configuring the locations and services of the modules in a cognitive system; the latter is a dynamic approach to discovering modules and their services, which eases task allocation and decomposition.

The *Data Representation* layer provides a common messaging subsystem (interchange format) that simplifies the exchange of information between modules by presenting a unified, shared vocabulary for the data. Heterogeneous and distributed systems naturally run into a "what language do you speak?" problem. Because the highly efficient transfer of data is critical to success in certain domains (e.g. radar countermeasures), we use a lightweight solution for data exchange in CORA to alleviate latency and ensure solution relevancy. While we chose an extensible mechanism that quickly and compactly encodes structured data while documenting its type, more heavyweight and human readable approaches may also be used.

The *Control Logic* layer is where the domain-specific "glue code" resides, exploiting the lower layers for information exchange and the higher layers for higher level reasoning processes and task autonomy. It uniquely specializes the CORA module to coordinate the deliberative and reactive processes, and translate data between the lower and higher levels.

The *Task Autonomy* layer is where (optional) agent-based problem solving paradigms fit in the CORA modules. Task decomposition and allocation functions are enabled by service discovery (messaging pattern layer), while goals, state information, and requests for problem solving help may exploit the data representation layer. A purely reactive agent is unlikely to make use of the (optional) knowledge layer, while a belief-desire-intention agent may make use of one or more reasoners (e.g. case-based and fuzzy) to support sophisticated autonomy in goal-based problem solving.

Finally, the *Knowledge Layer* is where one or more AI reasoning and representation techniques reside in a CORA module (see Section 2). Each module in an instantiated cognitive system making use of the CORA framework may employ none, one, or multiple of the reasoning approaches available in this layer. Hybrid reasoning is manifested by the use of multiple techniques both within and between individual CORA modules, as necessitated by the problem domain.

4. Application Examples

Part of the motivation for the CORA project comes from our participation on several projects which required multiple learning and reasoning paradigms working together, each supplying a part of the solution. In this section, we present three examples of such projects.

4.1 Cognitive Wireless Communications – BLADE

One example of work that demanded a hybrid approach is the DARPA BLADE project (Behavioral Learning for Adaptive Electronic warfare). GTRI participated on the DARPA BLADE project as part of a team with a number of subcontractors together providing multiple reasoning approaches integrated to address the larger problem. The DARPA BLADE project involved the detection, characterization, classification, and countermeasure generation of known and unknown wireless communication systems. Described below is the cognitive engine developed for BLADE, focusing on the Case-Based Learner and Reasoner (CBLR) which was supplied by GTRI. The CBLR consists of a two-phased approach: classification of the incoming signal, and selection and configuration of an appropriate countermeasure. Both phases use a case library built from a set of past examples in order to emulate the behavior of an expert performing similar tasks.

4.1.1 *Cognitive Engine Architecture*

Figure 3 (top portion) below shows a high-level view of the wireless cognition architecture. Optionally, Battle Damage Assessment (BDA) information can be used as feedback to inform the CBLR of the success or failure of a selected countermeasure. This can then allow updating feature weights or modifications to the case library. In instances where an adapted case is successful, the new case may be stored via online learning as a new case for reuse in future situations.

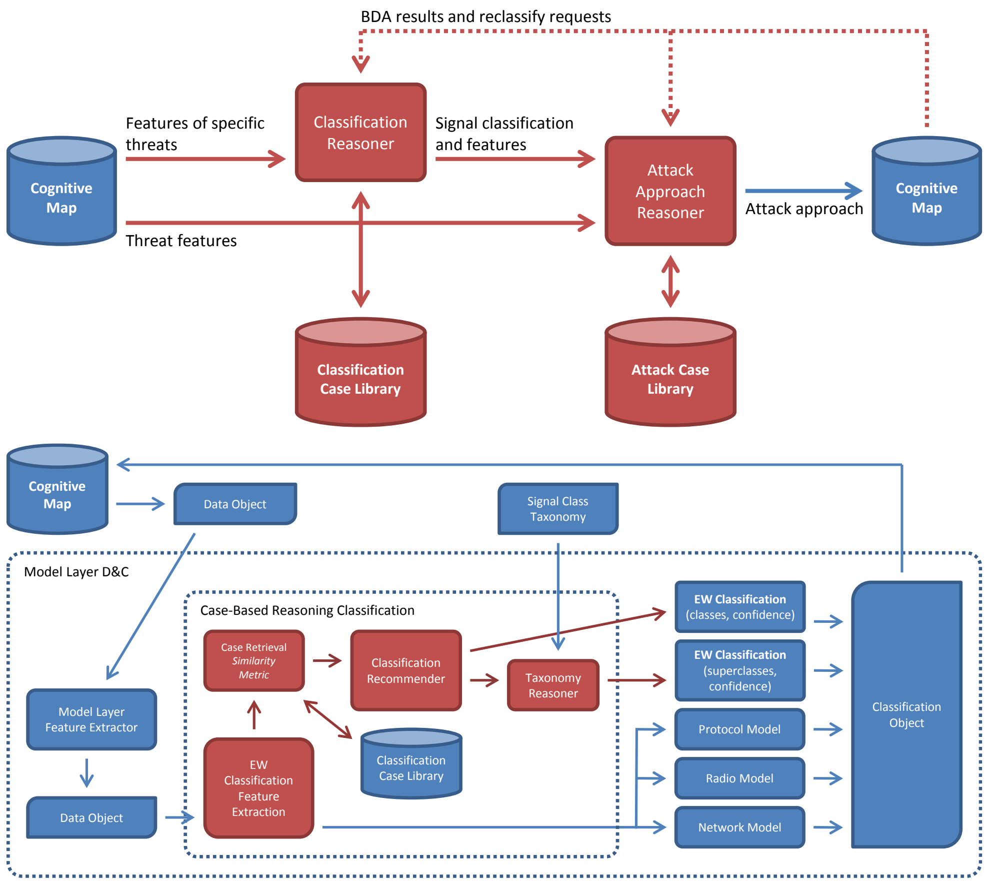
BDA results and reclassify requests
Features of specific threats
Classification Reasoner
Signal classification and features
Cognitive Map
Attack Approach Reasoner
Attack approach
Cognitive Map
Threat features
Classification Case Library
Attack Case Library
Cognitive Map
Data Object
Signal Class Taxonomy
Model Layer D&C
Case-Based Reasoning Classification
Case Retrieval Similarity Metric
Classification Recommender
Taxonomy Reasoner
EW Classification (classes, confidence)
EW Classification (superclasses, confidence)
Protocol Model
Radio Model
Network Model
Classification Object
Model Layer Feature Extractor
Data Object
EW Classification Feature Extraction
Classification Case Library

Figure 3. High level architecture depicting two stage reasoning for BLADE (top). Signal classification reasoner in BLADE (bottom).

Figure 3 (bottom portion) diagrams the knowledge containers, data flow, and reasoning processes used by the signal classification reasoner. The countermeasure reasoner uses a similar process. The model layer performs detection and characterization through four primary types of data objects that are stored in cases: Transmissions, Transmitters (transceivers), Channels, and Links. These objects contain attributes for parameterized threat models at four levels of a classification hierarchy.

This categorization scheme imposes a useful hierarchical structure on the classification and clustering of new feature data with respect to the generality and the computational effort required that serves as a guide for developing detection and classification (D&C strategies as well as countermeasures). Thus the case library is organized with similar hierarchies.

Most attributes entered into the objects are paired with intervals representing their observed range, i.e., their minimum and maximum values, which are supplied a priori from threat models. When comparing a target case against a historical case, the model layer compares the pairwise interval overlap of each feature to determine a feature-level distance metric. An interval A is defined as $A = [A_{start}, A_{end}]$. Once each feature's distance has been computed, an overall case similarity can be calculated.

The broader BLADE system (the integrated system with components from multiple team members) creates Transmission, Transmitter, Channel, and Link objects through a shared memory space called the Cognitive Map. These objects represent the individual components of an incoming signal at various levels, along with their relationships in time and frequency, among other features. As soon as a set of related objects is available, the CBLR uses a set of weighted similarity functions to compare attribute and parameter interval information (using a normalized Euclidian distance metric) of the newly-published objects to previous threat models stored in the case library in order to classify known threats. The set of feature weights are collected through subject matter expert interviews and refined through experimentation. A threshold parameter, when compared to the similarity of an existing case, indicates when to create a new case representing a new threat. This online learning is described in a later section. Once classification is complete, the CBR creates a classification report that summarizes the list of recognized threats along with any relevant metadata.

4.1.2 Solution Aggregation

The solutions of BLADE classification problems may contain multiple components. For example, a single Threat classification within a Classification Report may include both a *Threat Class* and a *Threat ID*, where "class" is a broader category applied to each known case and "ID" indicates an instance of a class. While the CBLR within the Model Layer supports training cases that define both of these solution parts together, it also supports training cases that have incomplete solutions. To use these partial cases, CBLR aggregates the solutions of multiple source cases when attempting to classify a target case. Aggregate solutions are created by executing multiple case retrieval cycles, storing any new solution components that are discovered as a result of each cycle. This process works as follows:

1. Perform kNN retrieval based on the target case
2. Execute the Taxonomy Reasoner to produce the best solution
3. Add any new solution components to the output solution.

4. If any components of the solution are missing, go back to step 1 with one distinction: retrieve only neighbors that contain new solution components.
5. Once all solution components have been populated, or kNN retrieval returns an empty set, return the output solution.

Solution aggregation allows the Model Layer to provide the most information for a given query. It is possible, however, that this process yields an output solution that is internally inconsistent, i.e. the components of the output solution are contradictory. This can be mitigated by providing the reasoner with more complete training case definitions prior to performance, since this will ensure that consistent solution components are retrieved together.

4.2 Electronic Sensing and Countermeasures for Novel Radar Waveforms

Another example of a project which required multiple learning and reasoning paradigms working together was a radar intelligence project. This project was similar to BLADE, except rather than spectral information representing communication waveforms, the spectral data involved radar. We used simulated radar data for our prototypes. Like the DARPA BLADE project, the novel radar waveforms project involved detection, characterization, classification, and countermeasure generation for known and unknown systems. By tailoring BLADE's case structure, CBLR system, and adding additional reasoning modules, we were able to apply our CORA architecture to reason in the new problem domain.

Figure 4 shows a diagram of our cognitive architecture, conceptually a two-stage process involving signal classification and characterization in the first, and countermeasure synthesis in the second. A radar signal (a time-varying sequence of electric "pulses") is received by an external clustering module, shown in green. This module generates "pulse descriptor words" (PDWs) to represent each pulse along with useful features, and then clusters them into groups representing sequences from distinct radar emitters. The clustered results flow into our PDW Feature Extractor, which creates statistical features that describe the entire cluster of PDWs; these extracted features are used by all other cognitive components, shown in blue, to determine a countermeasure based on mission parameters. The cluster results and extracted features are then passed to the CBR emitter classifier and the Bayesian Network modules. CBR is used to classify the set of PDWs by comparison to the emitter case library. The CBR emitter class is used by the Bayesian module for further classification of emitter type, mode, and intent. With classification complete, countermeasure synthesis begins.

The CBR and Bayes derived emitter classification results are used to perform countermeasure selection by CBR from the countermeasure case library. Adaptation techniques are used to transform the parameters of the retrieved countermeasure class to those that are applicable to the target case. Fuzzy reasoning is then used to perform additional countermeasure specialization. The countermeasure identifier and adapted parameters are delivered to the external countermeasure system for immediate activation in hardware. An explanation of the reasoning done to produce the countermeasure is provided for human consumption.

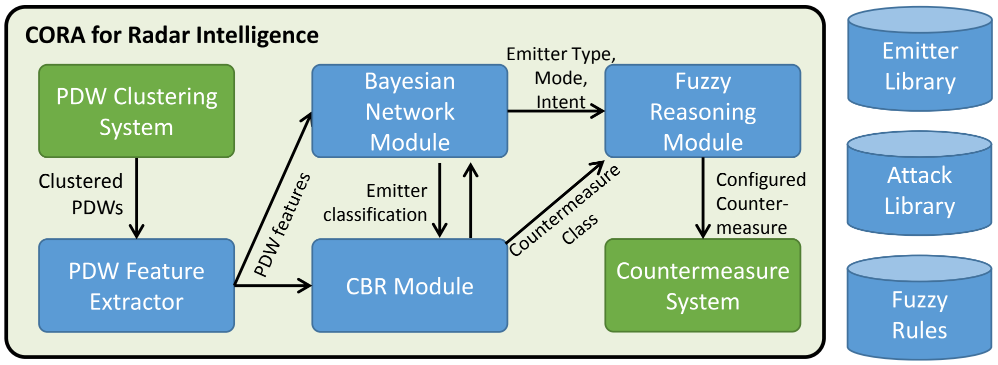

Figure 4. A cognitive architecture for performing ES and EW on novel radar waveforms.

4.2.1 Use of CBR for signal classification and countermeasure selection

Our Case-Based Reasoning (CBR) subsystem serves two functions. First, the subsystem uses CBR to determine the mode of the emitter by comparing features of the PDW cluster to a case library. Second, after other components of the reasoning engine have performed inference tasks, the CBR subsystem performs another cased-based analysis to determine the appropriate countermeasure for the emitter by comparing the emitter classification and the mission parameters to a library of known countermeasures. *Figure 4* depicts this dual role as first exchanging the emitter classification with the Bayesian network reasoner, which performs additional reasoning to make its own inference about the emitter type, mode, and intent of the emitter in question. The additional information is used to perform a new case retrieval to select an appropriate class of countermeasures. The retrieved case is passed to a fuzzy reasoner to adapt the countermeasure. The adaptation module is initially built to use knowledge from experts. As we test the system with simulated data we will evaluate the results and update the adaptation rules based on experimental results.

4.2.2 Use of Bayesian Networks for inference of emitter type, mode, and intent

The Bayesian Network (BN) module receives messages from the CBR Subsystem and the PDW Feature Extractor via the common messaging subsystem. When messages for a particular PDW cluster are received from both subsystems, the BN module uses a third-party Bayesian Inference Library to calculate probabilities for the radar type, objective and mode based on the supplied evidence. The inferred mode and objective are then published using the messaging subsystem.

The BN that we use to model this portion of the radar problem and perform inference is shown in *Figure 5*. The arrows in a BN represent the flow of influence in a model, not the flow of data. For example, the radar's mode determines the frequency, power and waveform emitted. We include nine variables. The domain of the emitter (e.g. air, ground) exerts probabilistic influence on the emitter type. The emitter type influences the radar's objective (e.g. scan, target, track), which in turn influences the emitter's current and previous mode. The mode of the emitter influences the carrier frequency, effective radiated power, the waveform class, and the CBR-determined mode. The inclusion of the radar's current and previous mode captures a Markovian relationship in emitter modes.

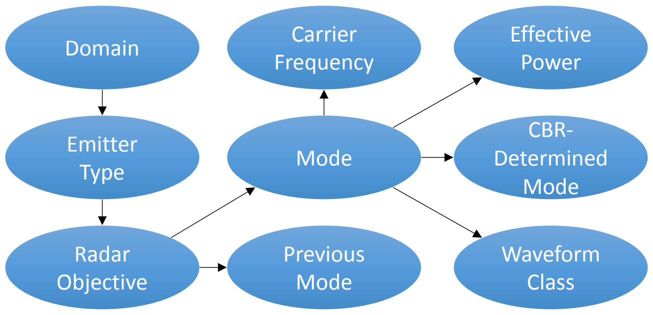

Figure 5. The Bayesian Network used to characterize emitters.

The inclusion of the CBR-determined mode in the model warrants further explanation. The CBR engine has a certain probability of correctly determining the mode of the emitter, which we capture as a directed edge from emitter mode to the CBR-determined mode. Once CBR has determined the mode of the emitter, the Bayesian network uses this input from CBR as a "strong hint" as to the correct mode of the emitter, but the Bayesian network may override the CBR-determined mode if other indicators are strong enough. This captures the idea that CBR may sometimes be confused by certain emitter parameters and may output the incorrect mode in these cases. When this happens, the Bayesian Reasoning Subsystem can help to correct the mismatch by considering the totality of the circumstances.

4.2.3 Fuzzy reasoning for countermeasure configuration

Finally, we take the selected countermeasure and all the other knowledge that was produced and use it to configure a countermeasure in the same manner as a human expert might. This is accomplished using Fuzzy Logic techniques, which enable the system to reason using imprecise concepts and dependencies. The major benefits to this approach include easier expert knowledge capture/modeling, easier maintenance, and a much smoother control surface.

There are variety of techniques that can be used to define Fuzzy Logic variables, rule sets, and functions. For our initial iteration, we chose to develop fuzzy sets and rules manually based off of the expert knowledge currently available to us.

In order to avoid a combinatorial explosion of our rule sets as each new dependency was added, we decided to make use of the Combs Method (Combs, 1998). The Combs Method adds some complexity in regards to less intuitive rules. However, it becomes increasingly attractive as the number of variables increases due to the reduced number of rules that need to be defined.

4.3 Intelligence Surveillance and Reconnaissance (ISR) Retasking

As another demonstration of our CORA architecture, we performed a simple knowledge acquisition and hybrid representation exercise in an Intelligence, Surveillance and Reconnaissance (ISR) retasking use case. Joint Publication 2-0 (2013) defines ISR as: "An activity that synchronizes and integrates the planning and operations of sensors, assets, processing, exploitation, and dissemination systems in direct support of current and future operations. This is an integrated intelligence operations function." ISR consists of separate elements but requires treatment as an integrated whole in order to be optimized. The experimental prototype depicted in *Figure 6* is designed as an iterative monitoring reasoner that receives inputs from the sensors and produces outputs for retasking and further situation awareness and decision making.

We are using a combination of CBR, Bayesian Networks, and Fuzzy logic to enable reasoning in the ISR retasking domain. In the ISR retasking domain, situational analysis and awareness drive much of the decision making. Therefore, we enable situational awareness by (a) reasoning over a taxonomy of resources (both friendly and enemy), such as time, money, material, (b) reasoning about the PMESII context (political, military, economic, social, infrastructure, and information), and (c) and providing retasking recommendations that respects constraints (e.g. terrain) and pre-specified utilities (e.g. rules of engagement, public relations, tolerance of casualties – civilian and troop). The output of the reasoning process is a retasking recommendation for gathering the most important missing intelligence, while respecting constraints (e.g. terrain, rules of engagement, public relations, and tolerance of civilian and troop casualties).

Consider the following scenario: A centralized control center receives streams of inputs from multiple ISR sources. The primary task is to ensure that the enemy does not ingress into the neutral zone. Sensors are used to gather missing pieces of intelligence to enable informed command decisions. Among the ISR assets are UAVs, UGVs, and unmanned watch towers. Towers can only tell vehicle type (air, ground, sea) and size (large, medium, small). UGVs move at 10 movement units, whereas UAVs move at 100 movement units. UGVs are heavily armored and slow, whereas UAVs are fast and can run for days without refueling.

To solve problems in this domain, we use CBR to retrieve cases containing Bayesian network fragments that represent small retasking information components; the retrieved Bayesian networks are used for determining vehicle tasking, based on the expected value of information; fuzzy reasoning is used for imprecise decisions (e.g. calculating reliable speed of a particular vehicle based upon distance, fuel remaining, mission priority, and terrain) and serves as a form of case adaptation. The output of the system is the evidence that should be used for retasking recommendation (e.g. to confirm that the unidentified vehicle is a tank, use drone 22).

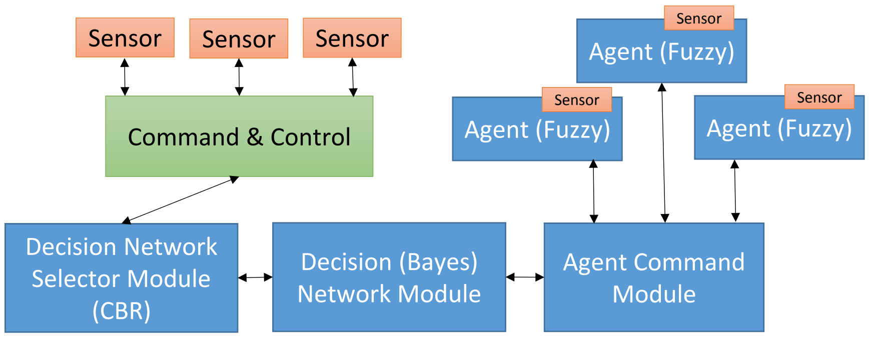

Figure 6. An instantiation of CORA for the ISR retasking problem

We first create and publish a situation awareness broadcast. This situation awareness broadcast is received by the ISR Retasking classifier (implemented as a case-based reasoner), which uses it to search its case base for a relevant case. The published case is received and correlated with the appropriate situation awareness broadcast by the Bayesian network reasoner, which uses the information to load the appropriate influence diagram. The network reasoner then performs a value of information calculation, which indicates it is either better to do take photos (assuming that doing nothing is not an option). The logger module in the background presents a single unified view of the messages and calculations as they pass through the system of systems.

The experimental prototype, while simple, demonstrated the value of integrating the multiple reasoners for robust decision making that includes many different kinds and sources of data. We are extending this system to include an explicit uncertainty reasoner, a meta-reasoner, and more distributable reasoning (as a multi-agent system). These extensions will support approaches to known problems described by our DoD customers such as Distributed Battle Management (DARPA) and Digital Commander's Intent (Air Force Research Laboratory).

5. Future Work

There exist numerous algorithms available to learn both the structure and parameter values of Bayesian networks, create more robust fuzzy sets and rules. Learning algorithm selection would ultimately be based on the structure, spatial attributes, and other properties of the data set. This is necessary given that certain algorithms have benefits and drawbacks for any given data set. We already have it within our capabilities to run a large variety of learning algorithms and techniques and anticipate applying them in future iterations of CORA.

In ongoing work, we are extending CORA to include more focus on the task autonomy, control logic, and shared memory layer pursuant to better supporting collaborative, distributed multi-agent problem solving. The example problems that we presented all rely on pre-allocated task assignments, even though the modules in some cases were distributed and capable of performing other responsibilities. Part of this work includes formalization of decisions as to when and why to use each of the learning and reasoning algorithms. We are designing experiments to compare the hybrid approaches both in terms of topology and interaction between the reasoning modules.

Most knowledge-based systems have uncertainty associated with data, reasoning, inferences, outcomes and decisions. Additionally, there are multiple sources of uncertainty that we wish to explicitly capture and reason about such as measurement errors, missing data, sensor data, the certainty of expert knowledge estimates, and errors introduced when combining results from hybrid subsystems. Inferences combining data from multiple sources with different uncertainties require a calculus for uncertainties combining results. Our treatment of uncertainty has so far been handled through adaptation knowledge, Bayesian networks for reasoning over likelihoods, and the distance between cases in CBR. We are currently prototyping a knowledge-based approach to reasoning about uncertainty that makes use of metadata to augment situation awareness data elements with uncertainty estimates, and include provenance and perishability information to improve reasoning. This approach relies upon associating metadata associated with data elements (their provenance, perishability and uncertainty) in order to allow explicit reasoning based these characteristics. Certainty factors have been successfully used in knowledge-based systems and are applicable to rule-based inferences and extendable to other types of inference.

6. Conclusions

The Cognitive Reasoning and Representation Architecture (CORA) is a flexible hybrid approach and toolset to enable human-like reasoning across a variety of problem domains. We have implemented and demonstrated a modular reasoning architecture including three reasoning modules capable of case-based, fuzzy and Bayesian network reasoning. We have also demonstrated three problem domains: cognitive wireless communications, electronic sensing for novel radar waveforms, and ISR sensor retasking. Future domains and problem types can be addressed through combination of existing reasoners and development of new modules within the CORA architecture.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological review, 111*(4), 1036.

Cassimatis, N. L. (2001). *Polyscheme: A cognitive architecture for integrating multiple representation and inference schemes.* Doctoral dissertation, Media Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Choi, D. (2010). *Coordinated execution and goal management in a reactive cognitive architecture.* Doctoral dissertation, Department of Aeronautics and Astronautics, Stanford University, Stanford, CA.

Combs, W. E. (1998). The Combs method for rapid inference. *The Fuzzy systems handbook*, 659-680.

Ferguson, I. A. (1992). *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents*. Doctoral dissertation, Clare Hall, University of Cambridge, UK.

Joint Publication 2-0, Joint Intelligence. *Defense Technical Information Center (DTIC)*. United States Department of Defense. 22 October 2013.

Kolodner, J. (1993). *Case-Based Reasoning.* San Mateo: Morgan Kaufmann.

Laird, J. (2012). *The Soar cognitive architecture*. MIT Press.

Langley, P. (2012). Intelligent Behavior in Humans and Machines. *Advances in Cognitive Systems, 2*, 3-12.

Pearl, J. (2014). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Trewhitt, E., Whitaker, E., Briscoe, E., & Weiss, L. (2011). Model Docking Using Knowledge-Level Analysis. In J. Salerno, S. Jay Yang, D. Nau, & S.-K. Chai (Eds.), *Social Computing, Behavioral-Cultural Modeling and Prediction* (Vol. 6589, pp. 105-112). Berlin: Springer.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & Sons.

Zadeh, L. A., Yuan, B., & Klir, G. J. (1996). *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A. Zadeh*. World Scientific Publishing Co., Inc.

Zhang, X., Yoon, S. W., DiBona, P., Appling, D. S., Ding, L., Doppa, J. R., ... & Whitebread, K. R. (2009). An Ensemble Learning and Problem Solving Architecture for Airspace Management. *Proceedings of Twenty-First Annual Conference on Innovative Applications of Artificial Intelligence, IAAI-09* (pp. 203-210). Pasadena, CA.