Visualization of Decision Processes Using a Cognitive Architecture

Mark A. Livingston[a], Arthi Murugesan[b], Derek Brock[a], Wende K. Frost[a], and Dennis Perzanowski[a]

[a]Naval Research Laboratory, 4555 Overlook Ave SW, Washington, DC, USA;
[b]NRC Research Associateship Award at the Naval Research Laboratory, 4555 Overlook Ave SW, Washington, DC, USA

ABSTRACT

Cognitive architectures are computational theories of reasoning the human mind engages in as it processes facts and experiences. A cognitive architecture uses declarative and procedural knowledge to represent mental constructs that are involved in decision making. Employing a model of behavioral and perceptual constraints derived from a set of one or more scenarios, the architecture reasons about the most likely consequence(s) of a sequence of events. Reasoning of any complexity and depth involving computational processes, however, is often opaque and challenging to comprehend. Arguably, for decision makers who may need to evaluate or question the results of autonomous reasoning, it would be useful to be able to inspect the steps involved in an interactive, graphical format. When a chain of evidence and constraint-based decision points can be visualized, it becomes easier to explore both how and why a scenario of interest will likely unfold in a particular way. In initial work on a scheme for visualizing cognitively-based decision processes, we focus on generating graphical representations of models run in the Polyscheme cognitive architecture. Our visualization algorithm operates on a modified version of Polyscheme's output, which is accomplished by augmenting models with a simple set of tags. We provide example visualizations and discuss properties of our technique that pose challenges for our representation goals. We conclude with a summary of feedback solicited from domain experts and practitioners in the field of cognitive modeling.

Keywords: Software visualization, visual knowledge representation, graph/network data, visualization in social and information sciences, cognitive model

1. INTRODUCTION

The objective of this research is to investigate the advantages of a defeasible, cognitively-based, graphic representation of decision making at various levels in a command structure. Our intent is to promote informed situational understanding by giving decision-makers a way to visually explore the component elements of a given decision context. To facilitate this goal, we employ cognitive modeling to represent the elements of a given decision process and construct an interactive visual display of the result. We envision a tool that allows evidence, tradeoffs, and alternative courses of action involved in a chain of decisions made by subordinates and/or automated agents to be quickly examined and weighed in terms of one or more desired outcomes.

As a basis for this effort, we argue that decision-visualization interfaces should meet three conditions of adequacy: the representation should be cognitively-based, readily accessible, and easily understood. Our rationale for using cognitive modeling constructs as the basis for showing how a particular course of action is likely to unfold is straightforward. To make adequately informed commitments in a given decision context, human decision makers need to be able to follow all of the situational reasoning involved. Cognitive modeling offers an intuitive and theoretically principled way to represent this specific class of knowledge. Moreover, as decision information in command settings is increasingly derived from a mix of human and autonomously generated sources, it is particularly important that cognitively-modeled accounts of the latter's procedures and performance parameters are instantiated to give decision-makers more confidence in courses of action that involve sophisticated forms of

Send correspondence to: Mark A. Livingston at mark.livingston@nrl.navy.mil, Telephone: 1 202 767 0380

automated reasoning. Secondly, information needed by decision-makers must be available at any point in the decision process. Knowledge of how a system reaches its conclusions and decisions, what constraints may have to be broken, and how one model compares to other alternatives is invaluable information. Thus, ideally, any point in a decision network should be inspectable by the decision-maker. Thirdly, the presentation of information must be sufficient to allow the decision-maker to easily comprehend how a decision process was achieved. This calls for the use of straightforward language and a visual display of the overall decision path. Furthermore, the level of detail and the manner in which information is presented will have a direct impact on that individual's confidence in autonomous systems. To facilitate the user's understanding, it should also be possible to interactively explore and alter the underlying assumptions. After reviewing related work, we describe how we use the Polyscheme cognitive architecture to generate information about the decision process and then construct a visual representation of it. We illustrate this with an example scenario.

2. RELATED WORK

We provide an overview of relevant and/or related work in two areas: the Polyscheme cognitive architecture and visualization techniques used for decision processes and related logical systems.

2.1 Cognitive Architectures

Computational theories of how human reasoning is organized, including ACT-R,[1] Soar,[2] and Polyscheme,[3] are commonly referred to as "cognitive architectures." ACT-R and Soar both have mechanisms for resolving impasses and competing rules, but unlike Polyscheme, which is employed in the work presented here, neither reasons in terms of alternative "worlds," which can be thought of as different models of truth for a given set of circumstances. In addition to this unique mode of operation, Polyscheme also employs a Bayesian belief framework and, thus, allows reasoning to be interpreted in terms of either costs or probabilities.

Knowledge pertaining to a given decision context is modeled in Polyscheme as a set of percepts and constraints. Percepts can be thought of as an initial body of declarative knowledge and are represented as atomic propositional relations, or "atoms," that hold for a set of entities over a specified period of time in a named world and have assigned truth values referred to as "stances." Entities can be constants or variables. Constraints express procedural knowledge and are represented as if-then rules that are structured to function as regularities in the world of the decision context being modeled, as well as in any of its alternatives. Constraints have three parts: an antecedent, a numerically-valued implication arrow, and a consequent. Antecedents and consequents are expressed as conjunctions of one or more atoms and, within a given constraint, represent a corresponding inference. Polyscheme functions as an inference engine that operates on a specified set of percepts and constraints, i.e., a "model," and produces a weighted set of alternative worlds. Polyscheme's stances on atoms in the world it determines to be the most likely outcome represent the architecture's "best" set of conclusions.

2.2 Visualizing Decision Processes

Few authors have directly addressed the issue of visualizing a decision process; thus, we review of visualization methods with more general approaches to similar types of information, noting some evaluations of these representations. Classical approaches to visualization of logical inference include Venn diagrams, Euler circles, and Peirce's alpha system.[4] Such systems encapsulate consequences, but do not allow for decisions to be inserted into the inference process. Similarly, standard node-link diagrams have been applied to scientific inference.[5] Again, there is no allowance for judgments to be explored or questioned in these approaches.

General Logic Diagrams (GLDs)[6] may be used to visually represent decision rules among many other logical constructs and functions. They use a table-based structure to capture the values of multiple variables; every cell represents a unique combination of values. Slate,[7] for example, relies on natural language processing of a controlled form of English among other data entry formats. It uses node-link diagrams to represent deductive reasoning arguments based on known facts, then extends these to probabilistic inferences. Nodes are either facts, inferences, or inference types (with numerical indicators of certainty). Directed links proceed from source (evidence) to conclusion. In a controlled study, users of Slate spent more time solving an illusory inference problem, but were more accurate than users of paper and pencil.

Graph inscribed logic (Grailog)[8] diagrams are directed labeled graphs that represent simple semantic hierarchies. Instances of constants or variables lead to nodes, whereas logical predicates (binary or n-ary) yield edges of the graph. Hatching patterns distinguish between the two node types, while line styles depict types of logical relationships (positive, negative, disjunctive, inclusive, et al.). Conjunctions may be represented implicitly by co-occurrence on the top-level of the hierarchy or by explicit appearance in a compound node. Disjunction can also be represented by a compound node (using stipple effects). General classes are usually represented by an oval-shaped node, instances are represented as rectangular nodes, and beliefs are represented by octagonal nodes (which may incorporate a relation among several instances). A logical relation is represented by a double arrow (similar to the mathematical symbol $\Rightarrow$) between the edges corresponding to the relations. DR-DEVICE[9] implements defeasible logic (a rule-based approach that allows incomplete and inconsistent information) with three representations: tree, a more general directed node-link diagram, and text. A user evaluation found that a hybrid approach (using all three) was superior to the tree representation, but the authors found no difference between the hybrid and the graph representations.

3. ALGORITHM

Although Polyscheme has previously been used as a framework for autonomous decision-making (e.g., see work by Trafton et al.[10]), to our knowledge, no effort has been made to give users a ready way to inspect the reasoning involved. To bring this information forward, we augment our models in Polyscheme with a specialized set of atoms (see Section 2.1), which, in turn, enables a graph-based representation of the decision process to be derived from Polyscheme's output.

3.1 Polyscheme Input

Our method for generating an interactive visualization of a decision process begins by encoding relevant contextual knowledge as a set of percepts and constraints in Polyscheme's formalism. Contingent percepts (and constraints) that can lead to different courses of action in the context are incorporated at this point by encoding alternative information and/or by changing pre-assigned stances for specific percepts in advance of a given run of the model (see Section 2.1). For example, in a model of traffic, a decision to make a right-hand turn at a red light can depend on which lane the driver is in, the presence of a "no turn on red" sign, or the driver's willingness to flout the law.

Each constraint with a role in characterizing any of the courses of action that can arise in the model is then augmented with a set of customized atoms, called "tags." Tags are used in the input file for the model to publish specific types of metadata about a constraint, including its categorical and contextual semantics, its identity, its logical structure (i.e., the rule's antecedents and consequents), and its inferential strength. Polyscheme reports its stance on all of these tags in its output, and this information allows an annotated account of the pattern of inferences and likelihoods that emerges in a particular run of the model to be constructed.

3.2 Extracting a Reasoning Process from Polyscheme Output

Polyscheme's output after it runs a model is a weighted set of alternative worlds (see Section 2.1). The alternative Polyscheme distinguishes as being the most likely corresponds to its best reasoning, and tags in the output for this world are used by our visualization algorithm to identify constraints Polyscheme instantiated. These rules and their matching metadata define a graph of Polyscheme's decision process. As an example, we illustrate this stage of our algorithm with a subset of metadata for the following two (paraphrased) constraints taken from a model of syntactic and semantic reasoning about the sentence *"The red car passed the black car."*

1. If (*antecedents*:) a passing event occurs, the event's agent is a car, and the event's object is a car, then (*consequents*:) the passing agent is a car and the passed object is a car.

2. If (*antecedents*:) a passing event occurs, the passing agent is a car, and the passed object is a car, then (*consequents*:) the passed object may be stationary or in motion.

```
Exists(Event#1, E, R)
Exists(State#2, E, R)
TAG(Event#1, Assignment, R)
TAG(State#2, State, R)
ConstraintStrength(Event#1, 1, World#1)
ConstraintStrength(State#2, 0.6, World#1)
```

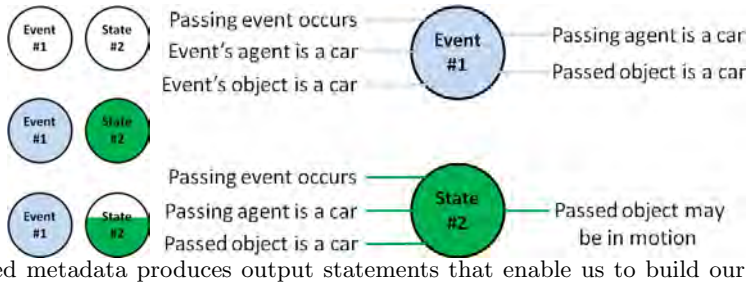

Figure 1. *Left*: Augmenting constraints with tagged metadata produces output statements that enable us to build our visual representation of a decision process. *Center*: Node generation starts with a contraint's existence and label (top row), continues with coloring by type (Event=blue, State=green; middle row), and finishes by setting its fill level to the corresponding truth value (here, 1.0 for `Event#1` and 0.6 for `State#2`; bottom row). *Right*: To create edges for nodes, we identify each node's component antecedents and consequents and then look for shared statements. Here, `Event#1` has consequents that the agent and object of the verb "passed" were both cars. These match two antecedents of `State#2`, so there would be an edge created from `Event#1` to `State#2`.

Although additional constraints are needed to reason about this sentence—for instance, the noun phrases that fill the roles of agent and object must be identified—these two rules are sufficient to illustrate the algorithm.

The algorithm first extracts nodes for the graph. When Polyscheme uses a constraint, we direct it to generate a set of tags as described in Section 3.1. This causes a range of metadata for the rule to appear in Polyscheme's output, including an existence statement, a type statement, and the constraint's strength with regard to possible outcomes in the decision context. The left side of Figure 1 shows these three tags for the two sentence-reasoning constraints in this example. In our modeling work, the strength of a constraint is treated a way to represent truth and uncertainty in the world and, thus, may be any number in [0,1]. The final entity (or parameter) in the two constraint strength statements shown on the lower left in Figure 1 gives the ID of the world in which these constraints had these strengths. As noted above, Polyscheme reasons through numerous alternatives and identifies the most likely world in its output. Our algorithm uses each constraint's existence, type, and strength tags to produce a progression of representations. A node is initially generated and labeled with an instance ID; it is then colored according to its type; and, last, if the constraint's strength strength is less than 1, its corresponding truth value is indicated by a corresponding fill level. This progression of steps is illustrated in the center portion of Figure 1.

The next step in the algorithm is the creation of edges between the nodes created in the first step. This is done by inspecting the tags for each constraint's antecedents and consequents that appear in Polyscheme's output. Paraphrases for the respective atoms corresponding to the antecedents and consequents in the sentence-reasoning constraints used in this example are shown on right side of Figure 1. For each match between any of the consequents of one node and the antecedents of another node, we create an edge from the node with the matched consequent(s) to the node with the matched antecedent(s). In this way, one can think of this process as discovering a dependency graph. By chaining together all of the constraints that were used in arriving at conclusions for the best world, we can understand the reasoning that led to those conclusions.

This simple example is missing one key element that we need in real scenarios: the ability for the user to make decisions. As illustrated in Section 4, we denote decision points with a diamond; this will indicate to the user where options are available that may lead to alternative outcomes that could be more favorable. Every input percept is a potential decision point, asking whether we believe that evidence is correct or not. Another element we detect is a set of states that exist on output—i.e., consequents with no matching antecedents.

4. RESULTS

We present a scenario that fits with our intended application, allocation of resources by law enforcement. In this hypothetical example, the resource is a patrol, and the decisions are where to assign that patrol and whether to believe two intelligence reports. We show the first four steps of the algorithm for extracting the graph, from creating nodes to the identification of matches between consequents and antecedents in Figure 2. Eight nodes are created, each from a different constraint. There are three node types: Intelligence (blue), Tactics (green), and

Figure 2. An illustration of our algorithm on a scenario that fits with our intended application. This shows the algorithm's progress through the matching of antecedents (leading into nodes from the left) and consequents (emerging to the right from nodes), just before the graph is built. The boxes are used for illustration of matches; this view is not shown to users.

outlaw Behaviors (lavender). Figure 2 is only for exposition of the algorithm. It shows the constraints involved in our scenario (represented as nodes) and demonstrates how directed edges between these nodes are defined by matches of consequents to antecedents.

We also created special nodes of type Violence for locations of interest; these are the outcomes that are most interesting for law enforcement in our scenario. Creating this special node type enables us to parse consequents that do not match with any antecedents and qualify as outcomes. This node is similarly filled according to the truth value assigned; we gave it a unique shape. The node definitions and properties along with the edges and their properties are translated into an input file for Graphviz,[11] which then draws the graph (Figure 3). We do a small amount of post-processing on the graph image, replacing nodes that we wish to have partly filled with a partial-fill rendering (a feature not available in Graphviz). Figure 3 shows three decisions (left side) must be made: whether to believe two intelligence reports and where to place the patrol unit. In this simulation, we conclude that Violence will occur in Atown with a probability of 0.94 and in Ctown with a probability of 0.50; these probabilities are represented by the proportion of fill in the corresponding nodes. Also shown in Figure 3 are three examples of tool-tips that we associate with nodes via tags. Note that one level of recursion (from the selected node, Behav#2) through the directed graph is depicted (cf. Figure 4).

In Polyscheme's parlance, changing a decision implies a change of the percept that encapsulates that fact about the world. When the user decides to send the patrol to the other location, three new nodes appear in the graph (Figure 4). The first is trivial; it merely states the existence of the patrol in the (newly) selected location. One can see that the constraint that generated this new node is the same as the constraint that was responsible for this decision in the previously computed world. Similarly, two other constraints are used with new arguments that make them come to different conclusions (probabilities for their outcomes). These nodes are added to the existing graph and the new edges are drawn; edges that are no longer valid are changed to dashed lines. The resulting graph shows both the initial world and the newly-computed world (Figure 4). There is still some chance that Violence will occur in Atown (right side of the graph; compare the two nodes labeled "Violence in Atown"). The patrol is no longer in Ctown; the original node for Tactic#3, which placed the patrol in Ctown in the initial world (Figure 4), is not part of the new world's graph (dashed lines only). Therefore, the new world (after the decision) has a new result from this tactical constraint and a slightly increased chance that Violence will occur in Ctown (node at bottom right versus node at top right).

Finally, on this more complex world, we can illustrate a useful case for the second interaction, which was specifically (and independently) suggested by a domain expert. We use a recursive search through the graph from an outcome to identify decision points that contributed to that outcome. The nodes and edges are highlighted through increased line width; decision nodes that are not involved are faded. See Figure 4 for an example.

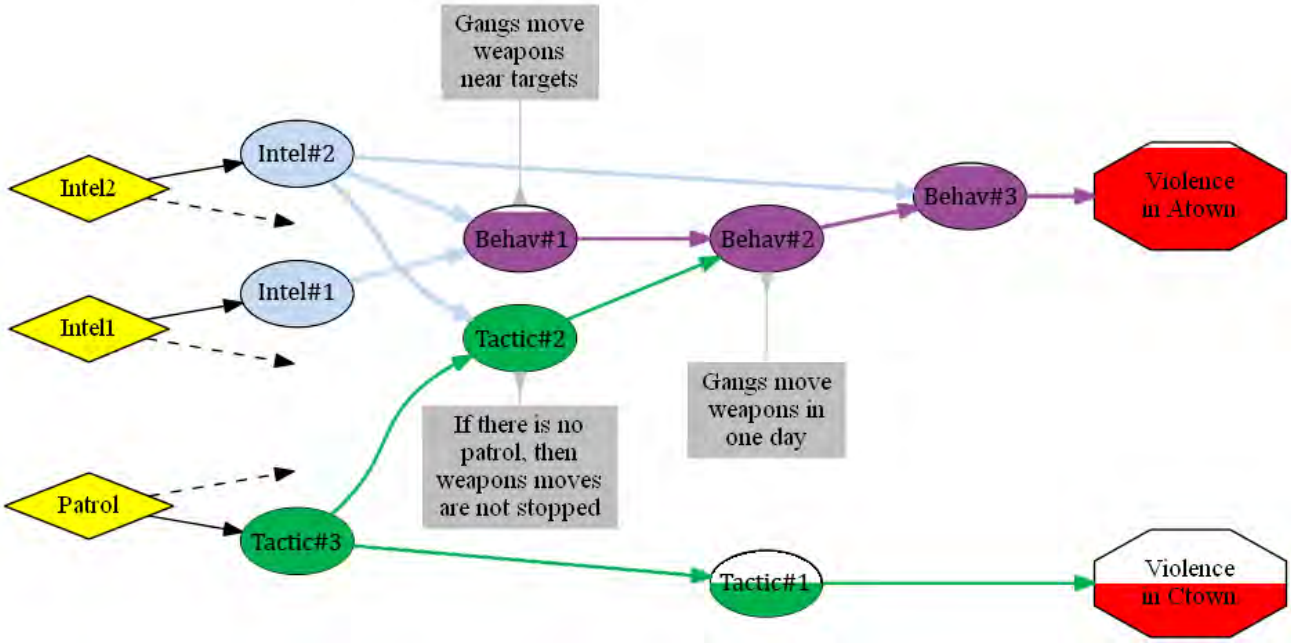

Figure 3. The final graph for the world computed from the input constraints. Three decision points (yellow diamonds) appear on the left; one conclusion (red octagon at upper right) is that Violence will occur in Atown with probability 0.94. As per the discussion of Figure 2, nodes establishing facts believed from Intelligence reports are light blue (not to be confused with yellow decision nodes that determine whether they are considered facts), Tactics nodes are medium green, and outlaw Behavior nodes are dark lavender. This image also illustrates the "tool-tips" that a user may use to explore nodes (here, with a single level of recursion).

One observation of these interactions is that the Graphviz tool does not have all the capabilities that we desire for our graphs. We perform a semi-automated post-processing of the resulting graph image in order to fix certain attributes. Nodes cannot be partly filled; this is a useful visual representation of uncertainty that we would like to use. Certain style attributes cannot be combined, such as the "bold" edges and filled nodes (which we fixed in post-processing, as seen in Figure 4). Filling or (partly) unfilling nodes is relatively easy, but would not be necessary with a graphing tool customized to our drawing guidelines. Another, more difficult challenge to overcome is the change in layout that results from introducing new nodes. Ideally, nodes that were in a graph before an interaction would not move when bubbles pop up to explain the reasoning at nodes or when new nodes are introduced by new constraints or alternative decisions. These issues cannot be overcome without a customized graphing tool, which we leave as future work.

5. DISCUSSION

Our method, though still a prototype, certainly meets the first criterion of adequacy that we identified (cognitive basis); the use of Polyscheme gives us this in our implementation. To assess whether the visualizations and interactive operations combine to meet the second and third criteria, we showed early versions of the visualizations of the scenario (including the alternative decision) described in the previous section to domain experts in cognitive architectures. (We added constraints after these expert design reviews. This forced Polyscheme to explicitly reason about Outcomes that were true by default, which in turn enabled our visualization algorithm to place them in the graphs in Figures 3-4. The recommendations from the expert users were unrelated to these changes.)

Our first expert was quite familiar with Polyscheme. His first comment was that he liked that we were able to avoid displaying much of the "extraneous" information that Polyscheme needs, but to humans is obvious – e.g. 17 February is within the month of February. Obvious as this is to humans, Polyscheme's processing of time requires explicit constraints developing relationships between days and months, and sequencing. A higher-level example is the proximity of two locations. To our end users (who will be familiar with the terrain), it will likely

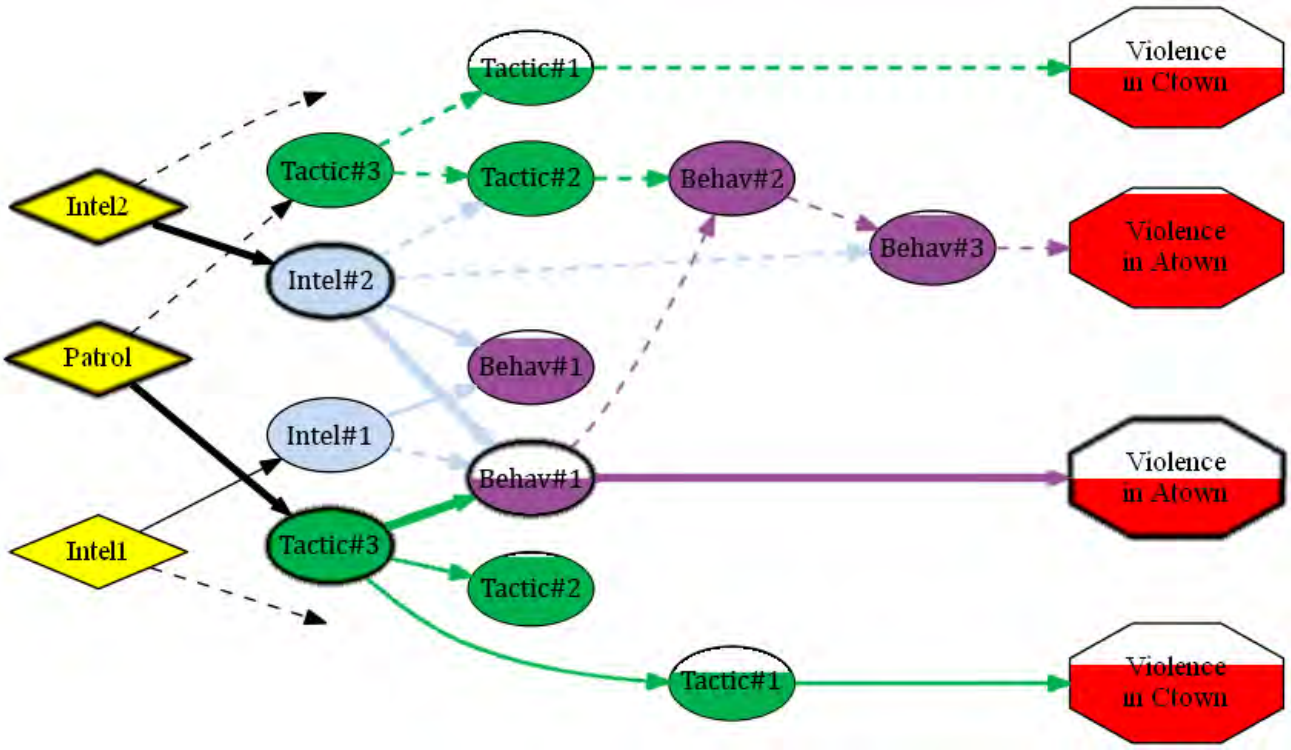

Figure 4. In the world that occurs with the new decision (assigning the patrol to Atown), there is no longer a high likelihood to Violence in Atown. However, there is a higher likelihood of Violence in Ctown, since the patrol is no longer there. New nodes in the graph appear generally in the bottom half. Edges that belong only to the first world (before the new decision) are dashed; nodes that belong only to the first world are adjacent only via such edges. New nodes derived from the same constraint receive the same label, although different truth values (probabilities) result. We also illustrate the recursive trace operation; two decisions contributed to the level of Violence estimated for Atown, shown with thick node borders and edges.

be common knowledge (as in our example) that Atown is far from Ctown. But Polyscheme currently requires a constraint expressing that the symmetric property applies to this relationship. This raises some deeper issues, however. If terrain status could be transitory during emergency scenarios, such as whether a road has been flooded and rendered impassable by a storm, then there may need to be awareness of terrain status pushed to users who are familiar with an area of operations. Still, the high-level views we provided without unnecessary details were viewed favorably.

Both of our experts were enthusiastic about the possibility of being able to view alternate scenarios. This was compared favorably with other cognitive architectures. ACT-R can display the global view of its system model at the end of its computations. This maps back to buffers that predict neural activations (with temporal sequencing). But this model is at a much lower level in the reasoning process, so the comparison is strained; our method of presenting alternate scenarios was viewed as a positive contribution that our system can make to the field. One could argue that our system needs more knowledge, since ACT-R has a wealth of neural models, whereas we must program Polyscheme with constraints. However, we get dependency trees and have far less-constrained worlds, while ACT-R gives a graph of the firing of neurons, but not really a causal chain.

One of our experts suggested that interaction would be critical. Several specific suggestions were made (before we showed the existing interaction operations). First, as a minor display change, users could specify short codes that could replace the numerical instance IDs that we currently display in the nodes. This may enable users to create mnemonics for actions or states that would assist with comprehension of the graph. Second, the recursive trace from outcomes to decisions was suggested as a useful interaction. (We then presented our design for this to the expert.) Finally, the expert recommended a tool that would highlight all the alternate worlds where a certain

outcome occurs. Then the user could examine those worlds for a pattern of decisions, to better understand the consequences of certain decisions. This alludes to the issue of how decisions may interact with each other in complex worlds where we ask Polyscheme to reason through uncertain information. We believe this will be a useful tool as we apply our implementation to more complex scenarios.

The final comment was a favorable view on our presentation of uncertainty at each node. This was considered reminiscent of Slate, which has strength factors in a $[-5, 5]$ range. Slate, which does not explicitly perform probabilistic inference, propagates the weakest strength factor among its antecedents to its inferred consequents. The strength of an inference is visually represented as text appended to the description of the node; for example, the strength 1 is described by the text "Probable" while strength 2 by "Beyond Reasonable Doubt." Our visualization of uncertainty, in which the amount of fill of a node indicates the probability, was favored over the textual descriptions.

One comment that we add based on our experiences is that this system as it exists now requires a fairly extensive programming effort in Polyscheme's modeling language. This is clearly not a task for the end-user, who is unlikely to be an expert in cognitive modeling in general or Polyscheme in particular. A tool to help the end user write constraints for the world would make the algorithm itself much more accessible.

Still, we believe that the comments from the experts in cognitive architectures support our assertion that we are moving towards meeting all three conditions of adequacy. Our methods distill the knowledge down to the non-obvious information and how it gets combined to reach conclusions about likelihood of outcomes. But we think that the graph representation of this reasoning that we have implemented is currently easily understood, and even with the enhancements suggested above, will remain comprehensible, while still enabling access to oftentimes complex reasoning involved in decision making.

ACKNOWLEDGMENTS

The authors wish to thank Nicholas Cassimatis, Jonathan Decker, Bruce Gudmundsson, Sangeet Khemlani, and Unmesh Kurup. This work was supported by the NRL Base Program.

REFERENCES

[1] Anderson, J. R., [*How Can the Human Mind Occur in the Physical Universe?*], Cognitive Models and Architectures, Oxford University Press (2007).

[2] Laird, J. E., Newell, A., and Rosenbloom, P. S., "Soar: An architecture for general intelligence," *Artificial Intelligence* **33**, 1–64 (Jan. 1987).

[3] Cassimatis, N. L., "A cognitive substrate for achieving human-level intelligence," *AI Magazine* **27**(2), 45–56 (2006).

[4] Allwein, G. and Barwise, J., eds., [*Logical Reasoning with Diagrams*], Studies in Logic and Computation, Oxford University Press (1996).

[5] Gooding, D. C., "Visualizing scientific inference," *Topics in Cognitive Science* **2**, 15–35 (Jan. 2010).

[6] Michalski, R. S., "Graphical minimization of normal expressions of logic functions using tables of the Veitch-Karnaugh type," *Journal of the Institute of Automatic Control* **52** (1978).

[7] Bringsjord, S., Taylor, J., Shilliday, A., Clark, M., and Arkoudas, K., "Slate: An argument-centered intelligent assistant to human reasoners," in [*Proceedings of the 8th International Workshop on Computational Models of Natural Argument*], 1–10 (July 2008).

[8] Boley, H., "RuleML/Grailog: The rule metalogic visualized with generalized graphs," in [*PhiloWeb 2011*], (Oct. 2011).

[9] Kontopoulos, E., Bassiliades, N., and Antoniou, G., "Visualizing semantic web proofs of defeasible logic in the DR-DEVICE system," *Knowledge-Based Systems* **24**, 406–419 (2011).

[10] Trafton, J. G., Cassimatis, N. L., Bugajska, M. D., Brock, D. P., Mintz, F. E., and Schultz, A. C., "Enabling effective human-robot interaction using perspective-taking in robots," *IEEE Transactions on Systems, Man, and Cybernetics* **35**, 460–470 (2005).

[11] Ellson, J., Gansner, E., Koutsofios, L., North, S., and Woodhull, G., "Graphviz – open source graph drawing tools," in [*Graph Drawing (Lecture Notes in Computer Science)*], 483–484 (2001).