In *Proceedings of the 2nd International Conference on Biologically Inspired Cognitive Architectures (BICA), 2011.*

Fusing Symbolic and Decision-Theoretic Problem Solving + Perception in a Graphical Cognitive Architecture

Junda CHEN,[a] Abram DEMSKI,[a] Teawon HAN,[a] Louis-Philippe MORENCY,[a]
David PYNADATH,[a] Nicole RAFIDI[b] and Paul ROSENBLOOM[a,1]
[a] *Department of Computer Science and/or Institute for Creative Technologies (ICT), University of Southern California (USC), USA*
[b] *Department of Electrical Engineering, Princeton University and USC/ICT, USA*

Abstract. A step is taken towards fusing symbolic and decision-theoretic problem solving in a cognitive architecture by implementing the latter in an architecture within which the former has already been demonstrated. The graphical models upon which the architecture is based enable a uniform implementation of both varieties of problem solving. They also enable a uniform combination with forms of decision-relevant perception, highlighting a potential path towards a tight coupling between central cognition and peripheral perception.

Keywords. cognitive architecture, decision making, graphical models, perception, CRF, SLAM, POMDP

Introduction

A *cognitive architecture* embodies a hypothesis about the fixed structure underlying intelligent behavior, whether in natural or artificial systems. Central to intelligent behavior, and thus to any such architecture, is an approach to *decision making*; i.e., to determining what actions should be performed as a function of what is perceived, what is known and what is desired. In a typical symbolic architecture, such as Soar [1], perception occurs via distinct perceptual modules interfaced to the architecture; knowledge takes the form of rules, facts, cases/episodes or general logical statements; and desires are encoded as goals. In a decision-theoretic architecture, knowledge takes a probabilistic form, with desires encoded as numeric utilities [2,3]. Due to uncertainty, many leading perception algorithms are likewise probabilistic, implying that they may be more compatible with a decision-theoretic formalism for decision making than a symbolic one; yet decision-theoretic architectures, to the extent that they do perception, also typically interface to separate perceptual modules [4].

The ideal architecture would leverage the combined strengths of both symbolic and decision-theoretic approaches while also tightly integrating perception into the overall framework. This article reports on the early stages of exploring such a fusion, via an architecture that leverages *graphical models* for hybrid (discrete and continuous) mixed

[1] Corresponding Author: University of Southern California, 12015 Waterfront Dr., Playa Vista, CA 90094, USA; E-mail: rosenbloom@usc.edu.

(symbolic and probabilistic) behavior [5,6]. Earlier work showed how such a graphical architecture could reproduce a standard form of symbolic decision making, based on how the Soar architecture uses long-term memory to generate candidate operators representing actions and evaluate them to yield preferences among them, and then selects among them via a preference-based decision procedure [7]. Soar also has the ability to interface with perceptual modules and to reflect when existing knowledge is inadequate for generating or evaluating operators; e.g., when evaluation knowledge is insufficient, it can engage in reflective search to determine which operators best reach the goal. This form of reflective search, which is central to symbolic problem solving, has also been demonstrated in the graphical architecture.

Here the focus is on incorporating into the same graphical models that underlie the architecture's memory and symbolic decision making: (1) a decision-theoretic approach to operator evaluation that is based on partially observable Markov decision problems (POMDPs); and (2) perception that probabilistically grounds decision making in the uncertain external environment. Out of the wide space of probabilistic perceptual algorithms, two are considered that combine state-of-the-art performance on dynamic multi-step perceptual problems with natural mappings to graphical models: conditional random fields (CRFs) [8] and simultaneous localization and mapping (SLAM) [9]. Although some work has been done on the mapping aspect of SLAM within the graphical architecture, the material here is limited to localization.

A set of experiments in a simple 1D navigation task is included to verify that the combined graph works as expected within the graphical architecture, and to begin exploring some of its resulting properties. But the main result here is qualitative: demonstrating the feasibility of fusing symbolic and decision-theoretic models of problem solving, while also coupling tightly with a uniform implementation of decision-relevant perception – to yield a secondary form of fusion between central cognition and peripheral perception – within a theoretically elegant graphical cognitive architecture. This pair of fusions contributes in two ways towards architectures capable of broad yet uniformly implemented and tightly integrated functionality.

The overall focus in graphical models is *computational* – specifically on efficient computation over complex multivariate functions by decomposing them into products of simpler functions – rather than *biological*, but graphical models do share many of the attributes of neural networks – performing limited forms of local computation on numeric messages within a graph-structured long-term memory – and a number of neural network algorithms map directly onto them [10]. There is thus an abstract form of biological inspiration, plus potential applicablity to more directly inspired work.

1. The Graphical Models

The backbone of each of the three techniques to be integrated with symbolic problem solving within the graphical architecture – POMDPs, CRFs and SLAM – consists of a chain of state variables over a sequence of time steps (Figure 1). The links between successive variables represent constraints over state transitions, which may encode transition probabilities – $P(X_i \mid X_{i-1})$ – or more general *potential functions* – $f_k(X_{i-1}, X_i)$ – whose values convey information

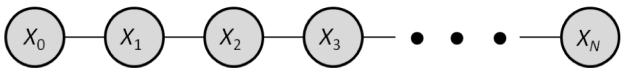

Figure 1: Graphical model (Markov network) for state transitions.

about the relative likelihoods of the transitions. POMDPs and SLAM use conditional probabilities while CRFs use potentials, but in either case the entire graph expresses a joint distribution, or function, over its variables that decomposes into the product of the individual distributions or functions. For POMDPs and SLAM, the graph computes $P(X_0,\ldots,X_N) = \prod_0^N P(X_i \mid X_{i-1})$, where X_{-1} is null in $P(X_0 \mid X_{-1})$ to yield the prior on X_0. CRFs omit priors and use potentials, but with the actual functions used in the product being weighted exponentials of the features, such as $\exp(\theta_k \bullet f_k(X_{i-1}, X_i))$. The overall equation for CRFs thus becomes $F(X_0,...,X_N) = \exp(\sum \theta_k \bullet f_k(X_{i-1}, X_i))$.

Attached to the states in this backbone are variables that represent observations of the world (Figure 2). The links connecting these observations to the states in which they occur – the ribs, to continue the metaphor – represent joint constraints over observations and states, expressed as further probabilities or potentials – $P(O_i \mid X_i)$ or $g_I(O_i, X_i)$ – that contribute to the overall product. This is how perception influences the state variables. Both CRFs and SLAM depend critically on observations across a sequence of states to jointly constrain the individual states. For example, in a CRF for word recognition that is based on observations of a sequence of letters, perceptions of all of the letters jointly constrain the identification of each individual letter. In a POMDP there is an observation for the initial state, but the later states in the chain are hypothetical during decision making, and thus do not involve observations.

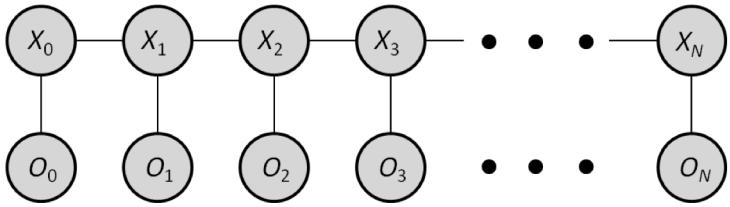

Figure 2: Graphical model (Markov network) for state transitions with observations.

For POMDPs and SLAM, action variables – representing operations that produce one state from another – are added to the transitions in this skeleton (Figure 3), extending the transition probabilities to $P(X_i \mid X_{i-1}, A_{i-1})$. The graphs in Figures 1 and 2 are Markov networks (aka Markov random fields), undirected graphical models over variable nodes, where there is a function implicitly defined over each pair of variables on a link. Markov networks can express functions over more than two variables, but this is awkward to display graphically. So we shift in Figure 3 to a more expressive type of graphical model that is central to the graphical architecture, *factor graphs*,

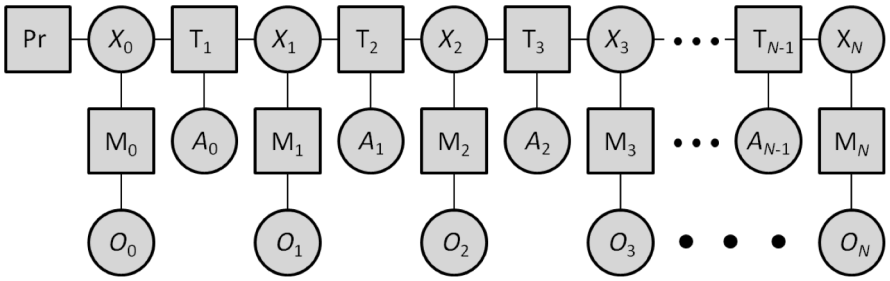

Figure 3: Graphical model (factor graph) for state transitions with observations (connected to states via mapping functions) and actions.

which include explicit factor nodes for functions. Like Markov networks, factor graphs are undirected, but factor nodes now explicitly represent the distributions or functions, and each connects to all of its variables. In the figure, factor nodes are represented as squares, with the T_i's transition functions, the M_i's map functions that relate objects to locations, and Pr the prior distribution on the initial state. All of the T_i's implement the same function, as do the M_i's.

To support operator evaluation via POMDPs, the graph must also include utility variables linked to future states, so that action distributions are based not only on the current state but on what is to be achieved (Figure 4). Essentially, constraint from localization flows forward in time while constraint from utility flows backwards. The overall result is effectively a dynamic decision network (DDN), although one in which decomposition of state variables – a significant feature generally in DDNs – has not so far been considered. This graph supports a different form of lookahead search from what is the norm in symbolic problem solving, and thus from what has been implemented already in a Soar-like manner in the graphical architecture. In the POMDP, the state variable at each time step represents a distribution over all possible states of the system at that step, possibly including the full combinatoric set of states in the problem space. The transition function represents a distribution over the states resulting from applying each possible action to each possible previous state. A single lookahead graph of a fixed length thus represents a probabilistic search to that depth.

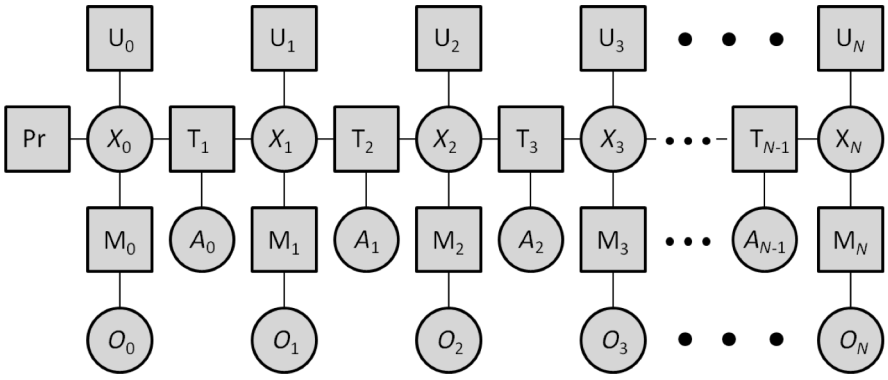

Figure 4: Graphical model (factor graph) with utility functions added for states.

Some years ago a distinction was introduced within Soar between *problem space search* and *knowledge search*, which maps onto various forms of dual process theory in psychology. Problem space search occurs across decisions. It is slow and serial, but provides an open-ended, potentially combinatoric search over an implicitly defined space, with the possibility of exploiting any available knowledge to help control the search. Knowledge search occurs within a single decision. It is fast and parallel, but can only search over a space that is explicitly defined by the existing memory structures, with no ability to use additional knowledge to control this search. The POMDP extends Soar's original notion of knowledge search to probabilistic lookahead that is combinatoric yet bounded by the memory structures. Still, as in Soar, when the evaluation knowledge derived from this knowledge search is insufficient to make a decision, there is a possibility of unbounded reflective problem space search.

The full graph that has been implemented (Figure 5) can be viewed as decomposing into three modules – one each for CRF perception, SLAM localization,

and POMDP action choice – that interact through shared variable nodes; although they are all actually implemented in a uniform manner within a single factor graph. The CRF and SLAM jointly concern the past and present, while the POMDP concerns the future. The CRF computes a distribution over the possible objects (O_i) at the current and previous locations (X_i) from sensations (S_i), using perception functions (sensation-object relations: P_i) and object-transition functions (object-adjacency relations: OT_i). Its backbone here is the perceived objects (O_i) rather than the locations (X_i), with the ribs being the sensations (S_i). There are three sensors (S^1-S^3), each with its own perceptual function (P^1-P^3). SLAM yields a distribution over the current location (X_0) from the object distributions (O_i) produced by the CRF and evidence about actions performed (A_i), while using map functions (object-location probabilities: M_i), movement-transition functions (probabilities of new locations given a location and an action: XT_i) and the prior (Pr), which is uniform in our experiments. The POMDP chooses an action (A_0) that maximizes expected utility given the current-location distribution (X_0) provided by SLAM, using utility (U_i) and movement-transition (XT_i) functions. In addition to the shared variables, the one other source of interaction among the modules occurs during the training of the CRF, which takes into account SLAM's map function (M_i) and SLAM/POMDP's transition function (XT_i).

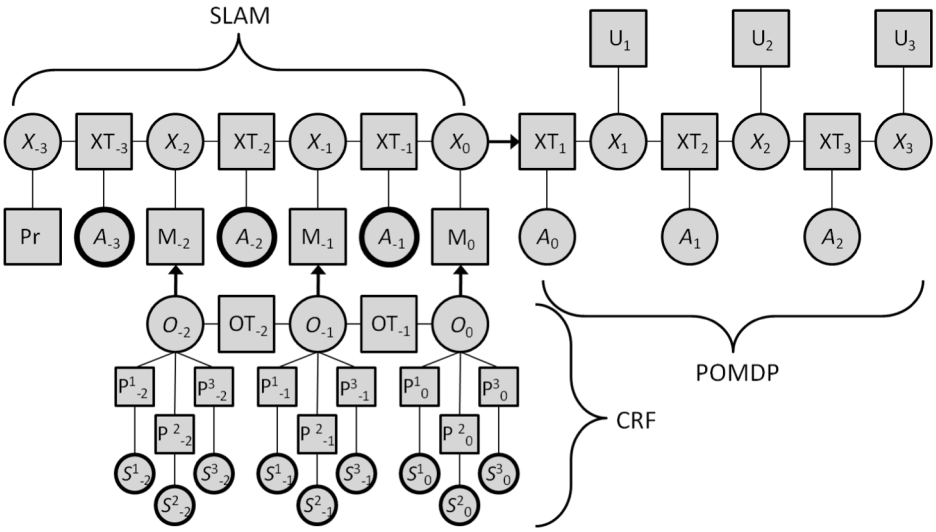

Figure 5: Complete graphical model (factor graph) for CRF+SLAM+POMDP. X_0 is the current state and A_0 is the action to be selected. Bold outlines indicate evidence nodes.

Each module involves bidirectional connectivity internally, and thus bidirectional sharing of information and uncertainty among variable nodes that are connected via factor nodes. The unidirectional connectivity shown across modules shouldn't be confused with the directionality in Bayesian networks. Here it concerns the flow of information – via messages passed by the summary product algorithm [11] – rather than the direction of probabilistic conditionality. By enabling flow from SLAM to the POMDP, but not in the reverse direction, the POMDP can exploit SLAM's localizations without the utilities used in POMDP's lookahead affecting localization.

Similarly, unidirectional flow from the CRF to SLAM implies bottom up perception without top-down feedback.

2. Implementation of a Movement Task in the Graphical Architecture

A discrete 1D movement task has been implemented, where the goal is to reach a specified location, and the actions move a step to the right or left, or do nothing (Figure 6). This is a simple task that was chosen primarily to demonstrate that the combined graph works within the graphical architecture; however, it does include real world complexity stemming from: ambiguity in what is perceived, such as which wall is seen when a wall is perceived; errors in the perception functions that can lead to recognizing the wrong object; and actions that do not always behave as they are specified. The resulting uncertainty concerning both what is being perceived and where the agent is located calls for the kind of evidence combination across steps provided by the CRF and SLAM, and the probabilistic decision making provided by the POMDP.

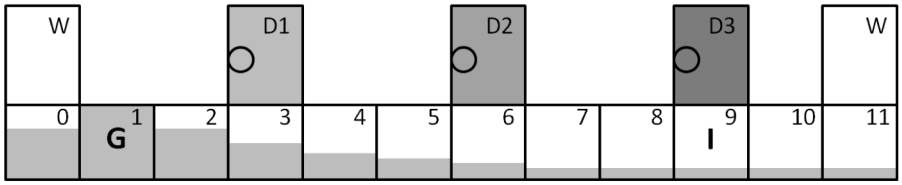

Figure 6: Task environment with two walls, three doors, an initial location (I) and a goal location (G). The relative values of the utility function for this goal location can be seen in the square shading. From the initial state, a lookahead of at least three is needed here before any discrimination is provided.

The graph in Figure 5 is solved at each step to choose the action to perform given the available evidence. The selected action is then performed in the world – an environment simulator – where it usually behaves as specified, but may fail. The graph acts as a sliding window that considers a fixed distance into the past (via the CRF and SLAM) and the future (via the POMDP). After an action is performed, the existing evidence is slid back a step (via extra-architectural code) and new observations arrive. The CRF senses rectangles (doors and walls), circles (doorknobs), and colors (doors have distinctive colors), usually correctly but occasionally in error. From the current and (within window) past sensations, object distributions are created for each time step. SLAM leverages these object distributions, along with evidence about previous actions, to generate a distribution over the current location. The POMDP exploits this distribution to initiate a probabilistic lookahead that yields a distribution over the action to be chosen. The best action is then selected in working memory, and applied in the world, enabling the whole process to repeat.

This task has been implemented within the graphical architecture without building specialized CRF, SLAM, and POMDP algorithms into it. Instead, *knowledge* is added to long-term memory (LTM) and *evidence* to working memory (WM). Since general perception and learning mechanisms are not yet in place, external code is used to initialize WM and LTM. LTM is encoded via *conditionals* built from *conditions*, *actions*, *condacts*, and *functions*. Conditions, actions and condacts are specified as patterns over named predicates with typed arguments. Conditions and actions behave much as in rule systems, with conditions matching to working memory and actions

proposing changes to it. Condacts – a neologism for *cond*itions and *act*ion*s* – meld these functionalities, by passing messages both from and to WM, to yield the bidirectional processing that is crucial in probabilistic graphical models. A conditional function defines a constraint over a combination of variables in the conditional.

Working memory includes a function for each predicate, with each compiling to a factor node to which variable nodes are attached. For each of the variable nodes in Figure 5 there is thus also an unshown WM factor node. Each conditional in LTM compiles to a factor subgraph, with across-conditional linkage based on common predicates, via sharing of WM nodes. The graphical architecture's compiler actually generates a factor graph that differs in some details from this nominal graph, even beyond the inclusion of WM factor nodes, but the two are logically equivalent. The graph shown, with three steps of input for localization plus a lookahead of three steps for action choice, compiles within the graphical architecture into 132 variable nodes and 161 factor nodes (for 293 nodes total). The largest graph with which we've so far experimented uses ten steps of input for localization and five steps of lookahead. It requires 376 variable nodes and 451 factor nodes (for 827 nodes total).

```
CONDITIONAL 'Transition_X1_X2_A1
 Condacts: (X1 location:x1)
           (X2 location:x2)
           (A1 action:a1)
 Function: 1<0,0,^>, 1<0,0,L>, .2<0,0,R>,
           0<0,1,^>, 0<0,1,L>, .8<0,1,R>,
           0<1,0,^>, .8<1,0,L>, 0<1,0,R>,
           ...
```

Figure 7: Location-transition conditional (^ denotes no-move action).

The bidirectional links in the figure arise from condacts, with the factor nodes defined by functions. Figure 7, for example, shows the conditional for the transition probabilities from location X_1 to location X_2 via action A_1. Conditions specify the unidirectional interface links among modules, as shown in Figure 8 for map M_{-1}, and are also used with actions to form rules that convert the results of operator evaluation into preferences for the next action, as shown in Figure 9. Selection in this context occurs via the same code that drives selection in declarative memory and symbolic decision making [7]. Behavior then occurs over a sequence of *graph cycles*, each of which involves passing messages within the factor graph – via a variant of the summary product algorithm that uses a mixture of integration and maximization to summarize out variables at factor nodes – until quiescence, and then selected changes being made to working memory.

```
CONDITIONAL 'Map_X-1_O-1
 Conditions: (O-1 object:o-1)
 Condacts: (X-1 location:x-1)
 Function: .8<0,W>, .2<0,^>, 0<0,{D1,D2,D3}>,
           .1<1,W>, .9<1,^>, 0<1,{D1,D2,D3}>,
           ...
```

Figure 8: Map conditional for objects and locations (^ denotes no object).

```
CONDITIONAL Acceptable
 Conditions: (A0 action:a0)
 Actions: (Selected operator:a0)
```

Figure 9: Action-selection conditional.

Three experiments have been run to verify that the combined graph works and to begin exploring its behavior. These vary: (1) the initial and goal locations, each from 0 to 11; (2) the localization length, from 1 to 10; and (3) the lookahead length, from 1 to 5. When parameters are not varying as part of an experiment, they are set to: the initial and goal locations, and the utility function shown in Figure 6; localization of 5 steps; and

lookahead of 3 steps. The localization subgraph is initialized with evidence for all prior steps that corresponds to what is sensed at the initial location, modulo noise, and the action of doing nothing.

The first experiment examined how the full graph worked across the space of problems, yielding overview data on its performance. The graph solved 78% of the 144 distinct problems within 30 cycles, with failures primarily due to the localization ambiguity resulting from empty starting locations, which all look alike. The solved problems required an average of 7802 messages/cycle and 14 seconds/cycle,[2] with an average ratio of 1.7 between the number of cycles to solve a problem and the minimum number of moves possible to solve it. Although the uncertainties and errors can thus lead to non-optimal moves, it usually recovers and solves the problem within 30 steps.

The second experiment evaluated the impact of varying the SLAM localization length from 1 (requiring 794 messages and 1 second per cycle) up to 10 (26447 messages and 34 seconds per cycle). Localization improved as the graph went from 1 to 6 steps, but then decreased from there. As the localization window got large, faulty observations and failed actions contaminated the localization process for too long (an issue that should be addressable by an incremental learning approach). The third experiment explored lookahead via the POMDP. Given the utility function plus the initial and final locations, a lookahead of 1 yielded random decisions, but anything more enabled it to head towards the goal. The strength of the correct action increased with lookahead length, as did the computational cost in terms of both messages per cycle (from 6621 to 9426) and time per cycle (from 11 to 20 seconds).

The key result here is that the architecture yields behavior corresponding to a combination of CRF, SLAM, and POMDP from knowledge-driven activities on top of the architecture's theoretically elegant, hybrid mixed model of memory and processing – based on factor graphs and the summary product algorithm – rather than from extensions to the architecture. The same memory and decision-making capabilities earlier shown to support things like semantic and rule-based memories, as well as symbolic problem solving, also yield decision theoretic problem solving and perception. The main issue in these experiments is the cost of processing the graph. A cognitive architecture must achieve ~50 msec per cycle to model real-time human-like results. The timings here are off by two to three orders of magnitude. However, further optimizations plus parallelizing message passing do look to provide a promising route.

3. Summary and Future

By investigating decision-theoretic problem solving (via a POMDP) in a graphical architecture that has already been shown capable of classical symbolic problem solving, a step has been taking towards fusing these distinct approaches. The uniform integration of perception (via a CRF), plus the localization it supports (via SLAM) for the POMDP, also demonstrates the additional potential for unification provided by this architectural approach, particularly between central cognition and peripheral perception.

Much additional progress is required beyond the step taken here. The POMDP work in isolation has shown that the graphical architecture enables a generic representation of time steps with a step variable – avoiding the need to replicate the

[2] The timings for the three experiments are from different machines. The exact values aren't critical; it is the orders of magnitude that are significant.

subgraph for each step – but this remains to be exploited for the entire joint graph. We are also looking to extend SLAM to mapping, the POMDP to multiagent reasoning and Theory of Mind [12], and the CRF to latent dynamic conditional random fields (LDCRF) [13]. It is further crucial to investigate how these graphs can scale efficiently, be learned, and integrate with other capabilities, such as: mental imagery, as a general intermediary between perception and cognition [14]; episodic learning and memory, or some other incremental approach, to utilize past observations in SLAM; and reflection, to cope with insufficient decision theoretic knowledge.

Acknowledgments

This work has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM) and the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AFOSR/AOARD). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

References

[1] J. E. Laird. Extending the Soar cognitive architecture. In *Artificial General Intelligence 2008: Proceedings of the First AGI Conference*, Memphis, Tennessee, March 2008. IOS Press.

[2] G. H. Ogasawara. *RALPH-MEA: A Real-Time, Decision-Theoretic Agent Architecture*. Technical Report No. UCB/CSD-93-777, EECS Department, University of California, Berkeley, 1993.

[3] R. C. Murray, K. Vanlehn, and J. Mostow. A decision-theoretic architecture for selecting tutorial discourse actions. In *Proceedings of the AI-ED 2001 Workshop on Tutorial Dialogue Systems*, 2001.

[4] T. Huang, D. Koller, J. Malik, G. Ogasawara, B. Rao, S. Russell, and J. Weber. Automatic symbolic traffic scene analysis using belief networks. In *Proceedings of the 12th National Conference on Artificial Intelligence*, pages 966-972, 1994

[5] P. S. Rosenbloom. Combining procedural and declarative knowledge in a graphical architecture. In *Proceedings of the 10th International Conference on Cognitive Modeling*, Manchester, United Kingdom, August 2010.

[6] P. S. Rosenbloom. Rethinking cognitive architecture via graphical models. *Cognitive Systems Research*, 12(2), 2011.

[7] P. S. Rosenbloom. From memory to problem solving: Mechanism reuse in a graphical cognitive architecture. In *Proceedings of the 4th Conference on Artificial General Intelligence*, Mountain View, California, August 2011.

[8] J. Lafferty, A. McCallum, F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, 2001.

[9] T. Bailey and H. Durrant-Whyte. Simultaneous localisation and mapping (SLAM): Part II State of the art. *Robotics and Automation Magazine*, 13:108–117, 2006.

[10] M. I. Jordan and T. J. Sejnowski. *Graphical Models: Foundations of Neural Computation*. MIT Press, Cambridge, Massachusetts, 2001.

[11] F. R. Kschischang, B. J. Frey, and H-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2): 498-519, February 2001.

[12] D. V. Pynadath and S. C. Marsella. PsychSim: Modeling theory of mind with decision-theoretic agents. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2005.

[13] L.-P. Morency, A. Quattoni and T. Darrell. Latent-Dynamic Discriminative Models for Continuous Gesture Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[14] P. S. Rosenbloom. Mental imagery in a graphical cognitive architecture. In *Proceedings of the 2nd International Conference on Biologically Inspired Cognitive Architectures*, Arlington, Virginia, November 2011. In press.