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

Mental Imagery in a Graphical Cognitive Architecture

Paul S. ROSENBLOOM[1]
Department of Computer Science and Institute for Creative Technologies
University of Southern California

Abstract. Can mental imagery be incorporated uniformly into a cognitive architecture by leveraging the *mixed* (relational and probabilistic) *hybrid* (discrete and continuous) processing supported by factor graphs? This article takes an initial step towards answering this question via an implementation in a graphical cognitive architecture of a version of the Eight Puzzle based on a hybrid function representation and a factor node optimized for object translation.

Keywords. Cognitive architecture, graphical models, mental imagery

Introduction

A *cognitive architecture* is a hypothesis about the fixed structure underlying intelligent behavior, whether in natural or artificial systems [1]. When combined with knowledge and skills, such an architecture should be capable of yielding effective performance across a diversity of domains. By definition, all cognitive architectures include mechanisms in support of central thought processes, minimally consisting of models of short-term and long-term memory along with an ability to make decisions. But it may also be elaborated to involve multiple forms of memory, complex decision making – based on problem solving or planning – learning, and other hypothesized capabilities.

The diversity of intelligent behavior supported in such architectures may be due to either their comprising a relatively large number of specialized mechanisms [2,3,4] or their supporting flexible interaction among a relatively small number of general mechanisms [5,6]. Both of these strategies have been followed in building integrated models of central cognitive processes, although the choice generally embodies an implicit tradeoff between ease of achieving broad coverage and theoretical elegance. When it comes to extending architectures beyond central cognition, with the addition of peripheral perceptual and motor capabilities, the former strategy has been the primary option, with distinct perceptual and motor modules being added to whatever mechanisms exist for central cognition. Even the Epic cognitive architecture, which has gone furthest in supporting end-to-end modeling of human performance – from perception through cognition and on to action – uses distinct perceptual and motor modules [7].

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

A variant of the latter strategy can be pursued via a dual level approach; implementing a small set of general mechanisms at a level below the architecture that can interact to yield a diversity of architectural mechanisms – spanning symbolic cognition and continuous perceptuomotor capabilities – at the next level up [8]. Two leading candidates for such a lower level are neural networks – as, for example, in [9] – and graphical models [10], with the work here based on the latter. Graphical models share neural nets' focus on local processing in a network of limited computational units, and some kinds of neural networks map directly onto graphical models, but the primary focus is computational rather than biological. The work here should thus be considered as loosely, or abstractly, biologically inspired, but as not so in the details.

As part of an ongoing exploration into combining broad coverage with theoretical elegance in a new form of cognitive architecture, the work here is based on a particular variant of graphical models – factor graphs [11] – that are capable of supporting *mixed* (relational + probabilistic) and *hybrid* (discrete + continuous) processing. Within central cognition, the mixed aspect of this implementation level has yielded a uniform memory architecture able to replicate capabilities normally part of distinct procedural rule-based memories and declarative semantic and episodic memories, plus a constraint memory that is not typically found in cognitive architectures [12]. Exploiting this mixed aspect, plus many of the mechanisms that were already implemented in support of the memory architecture, has also enabled the addition of basic problem solving capabilities [13].

The focus in this article is on extending the uniformity at the heart of this graphical approach even further, in the direction of the peripheral processing underlying perception and motor control, by including mental imagery. Those existing cognitive architectures that include mental imagery, such as Soar (version 9) [3], do so in a modular fashion, with memories and processes distinct from those employed in the more symbolic parts of cognition [14]. Here we'll leverage principally the hybrid aspect of the graphical approach, along with a specially optimized class of factor nodes, to yield the beginnings of a mental imagery capability that breaks down the barrier between the center and the periphery, yielding a more uniform and less modular approach to complete architectures that it is hoped will be extendable to the entirety of mental imagery, and even beyond, to perception – where graphical models such as hidden Markov models and Markov/conditional random fields already provide a wide range of state of the art models – and motor control.

The results to date focus on a version of the classic Eight Puzzle, but with the board represented internally as a multidimensional hybrid data structure. The key mental imagery operation required for this puzzle is *translation*, where tile locations in the plane are shifted as the tiles are slid from place to place. Section 1 reviews the graphical cognitive architecture, as composed of memory and problem solving capabilities. Section 2 then describes how the requisite representation and reasoning are implemented for the imagery aspects of the Eight Puzzle, while introducing a class of factor nodes specially optimized for object translation. Section 3 delves further into this approach to object translation, both to understand it better and to explore how far it might extend. Section 4 wraps up with a summary and future directions. The core contributions of this work are: (1) the demonstration of the feasibility of representing and reasoning about continuous mental imagery uniformly in a graphical cognitive architecture; and (2) the identification and optimization of a special class of factor nodes for imagery operations, which happen also to fill critical roles in support of other architectural capabilities such as reflection and episodic memory.

1. A Graphical Cognitive Architecture

The graphical cognitive architecture is based on factor graphs, which are similar to the more familiar construct of Bayesian networks except that they: (1) concern the decomposition of arbitrary multivariate functions into products of simpler functions, rather than just the decomposition of joint probability distributions into products of prior and conditional distributions; (2) utilize bidirectional links; and (3) include explicit factor nodes, in addition to variables nodes, representing the subfunctions into which the original function factors. In yielding bidirectional networks that can represent more than just probability distributions, factor graphs are actually more like Markov networks (aka Markov random fields) than like Bayesian networks, but they are still more general than the Markov alternative [15].

Inference in factor graphs typically occurs by some form of sampling or message passing. The graphical architecture is based on message passing, via a variant of the *summary product algorithm* that passes messages between variable and factor nodes concerning the possible values of the variables [11]. Roughly, incoming messages are multiplied together at all nodes, in a pointwise manner, with factor nodes also including their functions in the product and then summarizing out all variables not to be included in an output message to a variable node. When *sum* is used for summarization, the graph computes marginals on its individual variables. When *max* is used instead, the graph computes the maximum a posteriori estimation over all of the variables.

One of the prime determinants of the generality of a particular implementation of factor graphs is the representation used for factor functions and messages. The simplest implementations use tables with: one dimension per variable involved; a bounded number of discrete values along each dimension; and a functional value for each distinct combination of domain values across the variables. Although simple to define and process, such representations are limited to discrete functions; and even there their verboseness can yield major efficiency issues. An implementation focused on probability densities may instead be based on individual Gaussian functions – each specified by just two parameters – or more generally on mixtures of Gaussians. Such representations can be concise and effective in representing levels of uncertainty, but can become awkward in representing other forms knowledge.

The graphical cognitive architecture is based on multidimensional continuous functions that are specified as piecewise linear functions over rectilinear regions [16]. Such a representation can approximate arbitrary continuous functions as accurately as desired, while also supporting discrete and even symbolic dimensions. To implement a discrete dimension, the function's domain is unitized to integral values, such as the half-open intervals [0,1) and [1,2) for the integers 1 and 2. For a symbolic dimension, domain unitization is combined with a restriction on the function's range to Boolean values (0/1 for true/false). A symbol table is then also added – for use by researchers rather than the system – to map from unit domain intervals to symbolic labels. Table 1 illustrates this hybrid representational capability via a function over two dimensions, one of them

Table 1. Piecewise linear representation of the conditional probability of an object's weight given its category: P(W | C). The category is symbolic while the weight is continuous. Only a fragment of the full table is shown.

*w**c*	Walker	Table	…
[1,10)	.01*w*	.001*w*	…
[10,20)	.2-.01*w*	"	…
[20,50)	0	.025-.00025*w*	…
[50,100)	"	"	…

continuous and the other symbolic.

Boundaries among regions are introduced automatically, as needed, by splitting existing regions whenever a single linear function is no longer adequate for them. For example, if a function is currently 0 across its entire multidimensional domain, and the value at a single point is then set to 1, boundaries are added along each dimension to separate that point from all of the adjacent space. The value for the region containing that point is set to 1, while all of the regions now surrounding it are set to 0.

To introduce a boundary at a point along a dimension, an *n*-1 dimensional *slice* is added at that point that cuts across all of the other dimensions. In consequence, a function always comprises an array of rectilinear regions, as in Table 1. When a slice becomes unnecessary, because the regions on both sides of it now express the same linear function, the slice is automatically removed to maintain a minimal representation for the overall function. In the example above, if the value at the altered point once again becomes 0, the boundaries around it are removed.

In a Bayesian network, knowledge about joint probability distributions is stored in the network structure plus prior and conditional distributions over variables. Evidence concerning particular variables then constrains the values of these variables in the network before an inference algorithm computes the implications of the network over all of the variables given this evidence. A similar approach is taken in the graphical architecture. The factor graph embodies knowledge in its links plus in the functions stored in the factor nodes. A subset of the factor nodes comprise *working memory*, the internal cognitive state of the system. Working memory serves the role of evidence in the graph, fixing the values of particular variables during a single application of the summary product algorithm. Working memory may be changed between successive applications of the algorithm based on the results of graph processing.

The remainder of the factor graph comprises *long-term memory*, with the execution of the summary product algorithm serving to retrieve long-term knowledge into working memory. Fragments of knowledge in long-term memory are represented in the architecture via generalized *conditionals* that are compiled into subgraphs within the overall factor graph. Each conditional may contain *conditions*, *actions*, *condacts* and *functions*. Conditions and actions are like the corresponding structures in standard rule systems. Conditions match to working memory to constrain which rules fire and to determine variable bindings. Actions consult variable bindings from conditions to suggest changes in working memory. Condacts provide a combination of these two functionalities, yielding the kind of bidirectional processing required for correct probabilistic reasoning and for partial match in declarative memories. Functions are specified over subsets of conditional variables, and compile down to functions in particular factor nodes within the graph.

When conditions and actions alone are used, the resulting conditionals behave like standard rules. An accumulation of such conditionals can then serve as a procedural rule-based memory. Figure 1, for example, shows a conditional defining a heuristic rule that rejects any Eight Puzzle operator that moves a tile out of its goal location. When instead just condacts

```
CONDITIONAL GoalReject
 Conditions: (Operator id:o state:s x:x y:y)
             (Goal state:s x:x y:y tile:t)
             (Board state:s x:x y:y tile:t)
 Actions: (Selected - state:s operator:o)
```

Figure 1. Eight Puzzle heuristic that rejects from consideration operators that move tiles out of place.

and functions are used, the resulting conditionals behave more like fragments of traditional probabilistic Bayesian and Markov networks. Accumulations of such conditionals have yielded functionality akin to what is observed in declarative semantic and episodic memories, as well as the kinds of functionality provided by constraint memories. Conditionals based on other combinations of the four basic types of elements can yield hybrid and blended functionality. Figure 2, for example, shows a conditional defined via one condition, two condacts and a function (the one specified in Table 1). This provides a fragment of an extended semantic memory in terms of the conditional probability of an attribute given the concept.

```
CONDITIONAL ConceptWeight
 Conditions: (Object state:s object:o)
 Condacts: (Concept object:o concept:c)
           (Weight object:o weight:w)

 Function: [see table 1]
```

Figure 2. Conditional probability of the weight of an object associated with the state given its concept.

One key distinction over variables and their corresponding dimensions in the graphical architecture is between *universal* and *unique* variables [17]. Universal variables are like the variables in rule systems that yield all possible consistent bindings to conditions. Unique variables instead yield distributions over their best bindings, as generally desired when accessing declarative memories. For unique variables, only the single best value – as determined by aggregating over the factor functions in the relevant conditionals – is placed into working memory at the end of a cycle. This selection process serves as the core of the problem solving approach in the graphical architecture, enabling operators to be selected and applied based on knowledge in long-term memory [13].

2. Mental Imagery in the Eight Puzzle

The Eight Puzzle is defined in terms of a 3×3 board on which eight numbered tiles are placed and one cell is left blank (Figure 3). A legal move slides a tile that is horizontally or vertically adjacent to the blank cell into the blank cell. A problem instance consists of reconfiguring an initial board configuration by sliding tiles to yield a specified goal board.

Figure 3. The Eight Puzzle Board.

In the graphical architecture, the Eight Puzzle board is represented in working memory as a four dimensional hybrid factor function. Two of the dimensions represent the x and y extent of the board, with each defined to be continuous over the half-open interval [0,3). The other two dimensions are also numeric, albeit discrete, with one representing the problem solving state that the board is part of and the other a tile number in [0,9), with the 0 tile denoting the empty cell. These latter two dimensions could conceivably have been symbolic, but it was easier to use successive integers for the states and tiles rather than creating distinct symbolic names for them. Also, as will be discussed shortly, the numeric representation for states plays a key role in enabling the mechanisms developed for mental imagery to be reused in the implementation of a reflection capability within the architecture.

The representation of a board in working memory has a functional value of 1 for each four-dimensional region – comprised of an <*x*,*y*> extent, a tile, and a state – whenever that tile is in that region for that state. Otherwise, the region has a functional value of 0. Figure 4 shows a part of the representation used for the board configuration in Figure 3, with the state dimension omitted for simplicity and the locations of only the first two tiles indicated. The grey regions in the figure have a value of 1 while the clear regions have a value of 0, denoting that the center right cell – the square <[2,3),[1,2)> – is blank, while tile 1 occupies the top left cell: <[0,1),[0,1)>. The Eight Puzzle board and tiles fit quite easily within the rectilinear regions supported by the architecture, but more complex objects should also be representable as combinations of such regions.

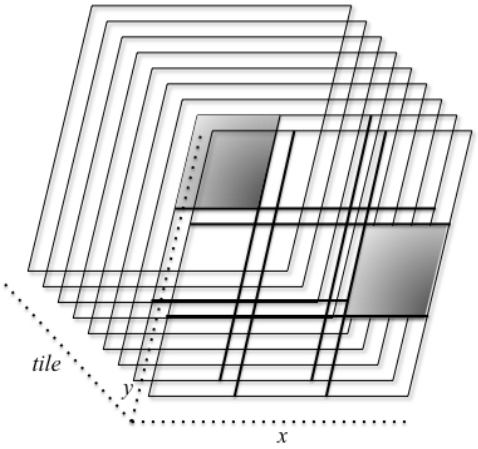

Figure 4. Partial visualization of three dimensions – two continuous and one discrete – of the hybrid representation used for the Eight Puzzle board.

Translating a tile from one cell to another requires shifting its location along either the *x* or *y* dimension. Because such operations on mental imagery involve two images of the board – from before and after the operation – plus a specification of the difference between them, it was earlier assumed that they would best be approached via the relatively complex kinds of graphical computations deployed in areas such as sequence prediction and stereovision [18]. However, a much simpler approach that is afforded by the piecewise linear structure of the image representation has so far proven sufficient. Given that altering the location of a slice automatically shifts the boundaries of all of the regions that abut the slice, the translation of a tile along a single dimension – which is all that is required in the Eight Puzzle – can be handled by offsetting the slices corresponding to its upper and lower boundaries along that dimension.

For example, Tile 1, which is at <[0,1),[0,1)> in Figure 4, can be moved to the right by shifting both of the slices that bound it along the *x* dimension – 0➔1 and 1➔2 – so that the tile is now located at <[1,2),[0,1)>. This is actually implemented via an action in a conditional that shifts to the right the entire plane corresponding to the 1 tile – altering the positions of all of its slices, not just those abutting the tile – with the region of the plane newly within the board at its left receiving a value of 0 and the region of the plane that is shifted off the right edge of the board cropped (Figure 5). Because each tile is in its own plane of the

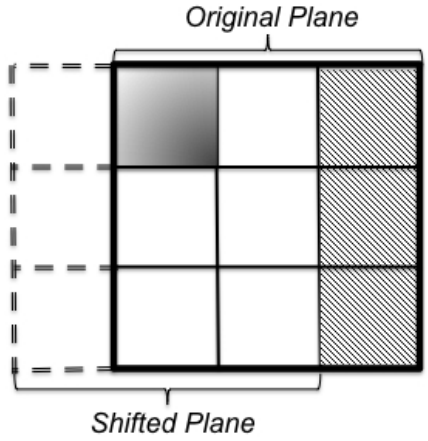

Figure 5. Translating the 1 tile to the right.

function, altering the location of one tile in this manner has no impact on the locations of the others.

Figure 6 shows the conditional that performs this shift of a tile to the right. It checks that an operator is selected (for the state) which is to move the tile located at <*x*,*y*>, and that the cell just to the right of <*x*,*y*> is blank. The first two actions shift the tile's plane to the right, by creating a new shifted plane and deleting the old one. The final two actions shift the empty cell leftward.

3. Offsets, Translation, and Beyond

The use of numeric offsets in conditionals induces one of the more intriguing aspects of this graphical approach to mental imagery. Translation occurs by specifying an offset for an action variable, as in the first and last actions in Figure 6. This leads to the insertion of a new factor node (and an associated variable node) into the action's subgraph that shifts the slices in incoming messages according to the offset (while properly handling new and cropped regions). In conditions, this same offset mechanism enables testing structures at positions relative to those of other structures. So, for example, the fourth condition in Figure 6 tests that the location just to the right of a tile is empty. It actually performs this test for all of the tiles, not just for the one that is to the left of the empty cell, but it only succeeds for this particular tile.

```
CONDITIONAL MoveRight
 Conditions: (selected state:s operator:o)
             (operator id:o state:s x:x y:y)
             (board state:s x:x y:y tile:t)
             (board state:s x:x+1 y:y tile:0)
 Actions: (board state:s x:x+1 y:y tile:t)
          (board – state:s x:x y:y tile:t)
          (board state:s x:x y:y tile:0)
          (board – state:s x:x+1 y:y tile:0)
```

Figure 6. Conditional to slide a tile to the right.

Beyond these two uses of offsets in mental imagery, offsets also play important roles elsewhere in the graphical architecture; in particular, in reflection and episodic memory. Reflection is being modeled on the Soar architecture [19]. When an impasse is detected in decision making – because a new operator cannot be selected – a new meta-level state is created in which reflective problem solving can be used to resolve this problem in the system's own performance. The discrete numeric representation used for the state then enables exploiting the offset mechanism to operate across states in the meta-level hierarchy. The contents of adjacent states, or even of states separated by some larger but still fixed distance, can be related via offsets in conditions, and information can be moved across states – to, for example, return results from reflective problem solving – via offsets in actions. Offsets in episodic memory enable, for example, accessing and comparing episodes that are adjacent, or at any fixed distance, along the temporal dimension. It thus turns out that a primitive architectural mechanism originally implemented for the translation of mental imagery not only provides this functionality – through its use in actions that modify imagery in working memory – but it also facilitates testing multiple relatively located portions of an image, as well as supporting key aspects of both reflection and episodic memory.

It is important to ask though whether such a mechanism fits properly into factor graphs, given that factor nodes normally compute subfunctions of the graph's globally

defined function over a subset of its full complement of variables; and they do so by computing the pointwise product of the messages arriving from these variables with its internal function, and then summarizing out all variables that are irrelevant to the outgoing message. It is also worth considering how far this mechanism might be extended both to additional aspects of mental imagery and to other architectural capabilities. These two questions are the subject of the remainder of this section.

Although shifting the locations of slices in piecewise memories is not a standard factor node operation, it is possible to encode translation in the normal fashion, validating that it does fit naturally into factor graphs. However, this is only possible in a manner that is expensive computationally, particularly for continuous dimensions. Consider first a simple discrete example, with a single dimension and a region of interest along the dimension that is restricted to [0,3). Suppose further that the functional values for the three regions along the dimension are <0, 1, 0> and that we want to offset this to the right, to yield the new vector of values <0, 0, 1> across these same three regions. If the vector <0, 1, 0> for variable *x* is sent to a factor node defined via the function in Figure 7, the resulting vector for variable *y* does become <0, 0, 1>, implementing the desired rightward shift.

The same approach works for any fixed offset. However, the approach engenders a function size that is proportional to the square of the length of the discrete span of interest (n^2); and regionalization can't compress this function because the diagonal layout of the 1s forces a fragmentation into unit regions. Even worse, offsetting a continuous dimension in this fashion would require a vast number of ε-sized regions along both *x* and *y*, with a resulting piecewise function of unwieldy extent: $(n/\varepsilon)^2$. By handling offsets directly – via slice shifting – translation occurs efficiently for both discrete and continuous dimensions. Thus, offsets define legitimate factor nodes, but ones that are best handled via a special purpose optimization, particularly for continuous dimensions. This turns out to be comparable to how negation is handled in conditionals, via a special purpose, more efficient, implementation of a distinct class of factor nodes [17].

2	0	1	0
1	1	0	0
0	0	0	0
y/x	*0*	*1*	*2*

Figure 7. Factor function for a discrete shift rightwards by 1.

A more abstract way of viewing the factor functions used to compute offsets in translation is as general delta functions – the Kronecker delta for discrete functions and the Dirac delta for continuous functions – as are more typically used for computing variable equality in factor graphs [20]. Delta functions are in fact already used in this manner in the graphical architecture, but there their efficient special-purpose implementation can be simpler, just amounting to message copying. Efficient representation of true delta functions in factor nodes requires a functional form more flexible than the rectilinear piecewise linear functions currently used in the architecture.

The Eight Puzzle only requires translation along one dimension at a time, but the offset technique works just as well for simultaneous translation along multiple dimensions, enabling object movement at any angle. It also appears that a conceptually simple extension to it will enable the additional mental imagery operation of *scaling* for continuous dimensions. Instead of adding a fixed value to the locations of the slices along a dimension, the slice locations instead need to be multiplied by a fixed scaling

factor. Each dimension could then conceivably be scaled independently by its own fixed factor, or scaling alternately could be restricted to a single multiplicative factor across all dimensions.

Rotation is the other major operation classically defined on objects in mental imagery. However, in contrast to translation and scaling, it cannot be performed independently along each dimension. Rotation not only conflates dimensions, but it also requires reslicing the image – as in Figure 8 for a rotated Eight Puzzle plane – to regenerate regions that are rectilinear along the original dimensions.

One potential approach for dealing with rotation is to allow more flexibility in slice orientation across multiple dimensions [16]. Should this work, it would not only enable rotation without major reworking of slices, but also more flexibility generally in the shapes of regions; allowing *convex polytopes* (*n*D convex polygons). Application of arbitrary combinations of translation, scaling and rotation would then occur via more general factor nodes capable of efficiently processing any form of *affine transformation*, comprising a linear transformation plus a translation. It is even conceivable that such an approach would eliminate the need for some or all of the specially optimized factor nodes, by enabling functions such as deltas to be specified via long narrow regions with values of 1 that are appropriately angled and offset.

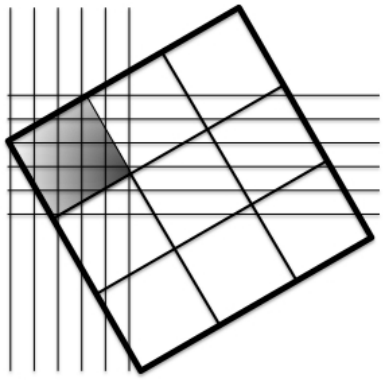

Figure 8. Reslicing a rotated structure.

4. Summary and Future

The work described here takes an initial step towards the uniform implementation of mental imagery with other cognitive processes in a graphical architecture. Imagery leverages the hybrid mixed function representation that is at the heart of the architecture plus the memory capability that is based on factor graphs. Specially optimized factor nodes for translation have been added that show potential for extension to a broader set of imagery transformations, and which also turn out to be critical for implementing several architectural capabilities that are outside of the normal scope of mental imagery. A more uniform approach to at least some of this may also eventually be possible through a more flexible region representation.

The focus in this article has been on two-dimensional imagery plus the translation operation, as needed for problem solving in the Eight Puzzle. With a total of 18 conditionals, a system has been constructed that can solve the Eight Puzzle by selecting and applying a sequence of translation operations defined over the hybrid board representation [13]. A longer term goal is to extend this nascent mental imagery capability to dynamic three-dimensional imagery capable of coping appropriately with uncertainty about the existence and trajectories of objects, as is necessary for example to support complex situation assessment and prediction. This will require many of the extensions already mentioned. It will likely also require: more sophisticated functional forms, an intimate connection with both perception and other cognitive capabilities, plus improvements in efficiency.

Acknowledgments

This work has been sponsored by the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AFOSR/AOARD) and the U.S. Army Research, Development, and Engineering Command (RDECOM). 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. I would like to thank Abram Demski for interactions that helped clarify some of these concepts.

References

[1] P. Langley, J. E. Laird, and S. Rogers. Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10: 141-160, 2009.

[2] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin. An integrated theory of the mind. *Psychological Review*, 111(4): 1036-1060, 2004.

[3] 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.

[4] B. Goertzel. OpenCogPrime: A cognitive synergy based architecture for artificial general intelligence. In *Proceedings of the 8th IEEE International Conference on Cognitive Informatics*, 2009.

[5] J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence,* 33: 1-64, 1987.

[6] M. Hutter. *Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability*. Springer-Verlag, Berlin, Germany, 2005.

[7] D. E. Kieras and D. E. Meyer. An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, 12: 391-438, 1997.

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

[9] Smolensky, P. & Legendre, G. (2006). *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar*. Cambridge, MA: The MIT Press.

[10] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge, Massachusetts, 2009.

[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] 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.

[13] 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.

[14] S. D. Lathrop, S. Wintermute, and J. E. Laird. Exploring the functional advantages of spatial and visual cognition from an architectural perspective. *Topics in Cognitive Science*, 2011. In press.

[15] B. J. Frey. Extending factor graphs so as to unify directed and undirected graphical models. In *Proceedings of the 19th conference on uncertainty in artificial intelligence*, pages 257–264, 2003.

[16] P. S. Rosenbloom. Bridging dichotomies in cognitive architectures for virtual humans. In *Proceedings of the AAAI Fall Symposium on Advances in Cognitive Systems*, 2011. In press.

[17] P. S. Rosenbloom. Implementing first-order variables in a graphical cognitive architecture. In *Proceedings of Biologically Inspired Cognitive Architectures 2010: Proceedings of the First Annual Meeting of the BICA Society*, Arlington, Virginia, November 2010. IOS Press.

[18] P. S. Rosenbloom. Speculations on leveraging graphical models for architectural integration of visual representation and reasoning. In *Proceedings of the AAAI-10 Workshop on Visual Representations and Reasoning*, 2010.

[19] P. S. Rosenbloom, J. E. Laird, and A. Newell. Meta-levels in Soar. In P. Maes, D. Nardi (eds.) *Meta-Level Architectures and Reflection*, pages 227-240. North Holland, Amsterdam, Netherlands, 1988.

[20] H-A. Loeliger. An introduction to factor graphs. *IEEE Signal Processing Magazine*, 21(1): 28-41, January 2004.