Extending Mental Imagery in Sigma

Paul S. Rosenbloom

Department of Computer Science & Institute for Creative Technologies
University of Southern California
12015 Waterfront Drive, Playa Vista, CA 90094
rosenbloom@usc.edu

Abstract. This article presents new results on implementing mental imagery within the Sigma cognitive architecture. Rather than amounting to a distinct module, mental imagery is based on the same primitive, hybrid mixed, architectural mechanisms as Sigma's other cognitive capabilities. The work here demonstrates the creation and modification of compound images, the transformation of individual objects within such images, and the extraction of derived information from these compositions.

Keywords: Mental imagery, cognitive architecture, graphical models, piecewise continuous functions, affine transformations.

1 Introduction

Mental imagery is a cognitive capacity that enables humans to represent and reason about spatial information. It includes the ability to construct images from pieces retrieved from memory; to translate, scale and rotate (parts of) these images; and to extract new information from the composite and/or transformed results. Although nominally focused on the spatial aspects of the physical world, Gunzelmann and Lyon summarize its key role in other areas of human cognitive processing – such as numerical information processing, problem solving and language [1] – and Cassimatis has hypothesized that physical reasoning is part of a general cognitive substrate that underlies all of reasoning [2]. Mental imagery must also clearly relate to perception, but the focus here is on the connection with cognition rather than perception.

Following an extended debate concerning whether mental imagery is symbolic versus imagistic – based, for example, on pixel arrays – there is little doubt at this point that both are implicated in the full picture. Some of the most interesting recent work on this topic includes how to incorporate such a capacity into a *cognitive architecture*, a hypothesis about the fixed structure underlying cognition, and how these structures combine with each other (and knowledge) to yield intelligent human(-like) behavior. Imagery modules have been investigated in architectures such as Soar [3] and ACT-R [4], including ideas for introducing more explicitly hybrid aspects [5].

The *Sigma* (Σ) architecture is built to be hybrid from the ground up, in service of satisfying two general desiderata: *grand unification* and *functional elegance*. A traditional unified cognitive architecture attempts to bring together in an integrated

manner the range of cognitive capabilities required for human(-level) intelligent behavior in the world. A grand unified architecture goes beyond this, in analogy to a grand unified theory in physics, to attempt to include the crucial pieces missing from a purely cognitive theory, such as perception, motor control, and emotion. Functional elegance implies a combination of the broad range of capabilities required in a (grand) unified architecture with simplicity and theoretical elegance. In Sigma, the aim is something like a set of *cognitive Newton's laws* that yield the required diversity of behavior from interactions among a small set of very general primitives. Within AGI, AIXI [6] can be seen as an attempt at an extreme form of functional elegance. The approach in Sigma is less ambitious, but still strongly in this direction.

Driven by these desiderata, work to date on Sigma has been deliberately broad – including forms of memory and learning [7-8], problem solving and decision making [9-10], perception and localization [10], and natural language – with the intent of determining whether a small set of general mechanisms can in fact be sufficient in combination. Thus, for mental imagery the natural question to ask became whether Sigma could provide a sufficient hybrid capacity without either distinct symbolic versus imagistic modules or distinct representations, memories and processes, as has been necessary in other architectural approaches.

Earlier work in Sigma showed how 2D images can be represented, and how translation of image components can be implemented [11]. The results were used as part of a hybrid approach to the Eight Puzzle – Fig. 1 – a classic sliding tile puzzle that is traditionally solved in AI systems via a symbolically represented board plus internal search over symbolic operators that model external actions. The Sigma approach included a hybrid representation of the board plus normal symbolic problem solving, but now over imagistic tile translations (implemented as *offsets*). The work here extends this via manipulations of *Z tetrominos* (Fig. 2), as found in the game of Tetris, to demonstrate: image composition and component deletion; additional forms of image transformation, including scaling, reflection and rotation (by multiples of 90°); and extraction of perceptual features from composites, such as object overlaps and collision detection, directionality among objects, and edge detection.

Fig. 1: Eight Puzzle.

Fig. 2: Z tetromino.

Mental imagery in Sigma is grounded in: (1) the architecture's generalized language of *conditionals*, which compiles down to *factor graphs* for processing via the *summary product algorithm* [12]; (2) an inherently continuous *piecewise linear representation* for the functions and messages in (1) [13]; and (3) *affine transformations* – a generalization of the offsets introduced earlier – and *piecewise linear filters*. By demonstrating mental imagery via interactions among more primitive mechanisms, this work contributes to the breadth of functionality unified within Sigma, while doing so in a simple and elegant manner. The key to functional elegance here has been to begin with a small set of very general mechanisms that are leveraged in combination when possible, and which are (minimally) augmented when necessary. This combination also supports grand unification, intertwining continuous perception-related information with general symbol processing.

2 Sigma and Mental Imagery

Knowledge representation in Sigma is based on *conditionals* – a generalized form of rule – plus *piecewise linear functions*. Fig. 3, for example, shows a conditional that uses two conditions and an action to determine the spatial overlap between two particular objects in the image. The `Image` predicate specifies object locations via three arguments – `o` provides a numeric index into a vector of objects (which is specified by constants here, but can be variables in general), while `x` and `y` range over the continuous image dimensions – to yield a discrete vector of continuous planes, each element of which provides an occupancy grid for a single object in the image. In Fig. 4, for example, the planes correspond to the Eight Puzzle tiles, and the grayed regions denote where the blank and tile 1 are located in Fig. 1.

```
CONDITIONAL Overlap-0-3
   Conditions: (Image o:0 x:x y:y)
               (Image o:3 x:x y:y)
    Actions: (Overlap i:1 x:x y:y)
```

Fig. 3: Conditional for computing the spatial overlap between two specific objects.

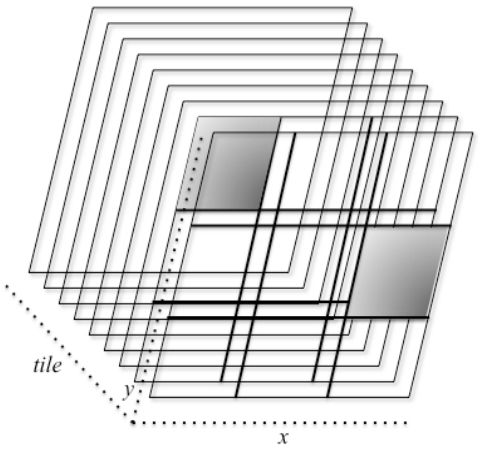

Fig. 4. Partial visualization of a hybrid representation for the Eight Puzzle board.

Technically, such images are 3D hybrid functions in Sigma, with one discrete variable (i.e., dimension) for objects (tiles here) and two continuous variables for space. The grayed regions are where the function has a value of 1 (or *true*) while the other regions are 0 (or *false*). In Sigma such functions are represented as piecewise linear over *n*D arrays of rectilinear (or *orthotopic*) regions (Fig. 5). The *n*D space is *sliced* orthogonally to its axes to generate an array of regions that are doubly linked along each dimension, with each region having its own linear function over the variables. This can be viewed as a generalization of a pixel (or voxel) array, where the pixels can vary in size and have linear rather than just constant value functions.

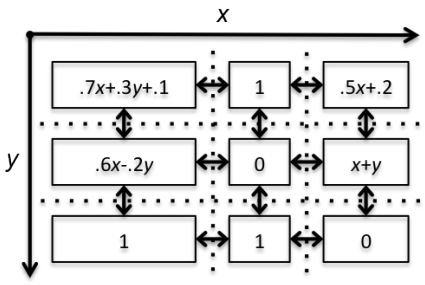

Fig. 5: Bivariate function as a 2D array of regions with linear functions.

Although Sigma's function representation is inherently continuous, with its piecewise linear approach allowing arbitrary continuous functions to be approximated as closely as desired, it

can also be specialized to: discrete representations – to enable, for example, the vectors of objects we have seen, as well as discrete probability distributions – by mapping integers in the function's domain to unit regions; and symbolic representations by limiting the functions to Boolean (0/T and 1/F) while assigning symbols to domain integers. A form of *hybrid mixed* representation is thus proffered in a manner analogous to how digital circuits are implemented via restrictions on an underlying substrate that is naturally continuous.

The `Overlap` predicate in Fig. 3 is similar to the `Image` predicate, except that here there is a vector of object overlaps, rather than of objects. The semantics of conditionals specifies that the two conditions in Fig. 3 extract the 0th and 3rd objects from the image, with these two 2D object subimages then being multiplied in a *pointwise* manner – like an inner product but without the final summation – to yield a new 2D plane that is 1 where both input planes are 1 and 0 elsewhere (Fig. 6). Once message passing reaches quiescence, actions – such as the one in Fig. 3 that stores the computed overlap into element 1 of `Overlap` – may yield changes to working memory, completing Sigma's core cognitive cycle. If a predicate has a *unique* variable – akin to a classic random variable, where a distribution is provided over all possible values but only one is ultimately correct – the best value for that variable is placed into working memory, while if it instead has only *universal* variables – akin to classic rule variables, where any subset of the values may be correct, but used here mainly for occupancy grids – all non-zero values are placed in working memory 14].

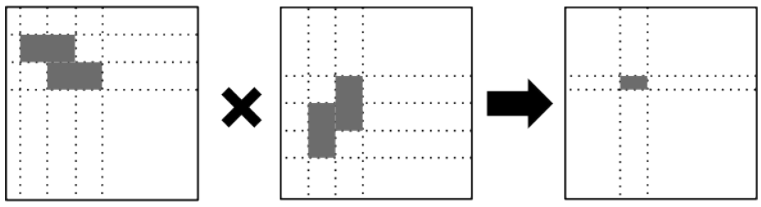

Fig. 6: Overlap between two images via conditional in figure 3.

The processing of conditionals occurs by running the summary product algorithm over the factor graph into which they are compiled. Factor graphs are a general form of *graphical model* [15] – an approach to computing efficiently over complex multivariate functions by decomposing them into the product of simpler factors and then mapping the result onto a graph of nodes and links – that bear a family resemblance to other forms, such as Bayesian and Markov networks, but are concerned with arbitrary multivariate functions, not just probabilistic ones. Complex functions are first decomposed into products of simpler functions, and then mapped onto bipartite graphs, with variables mapped onto variable nodes and decomposed factors mapped onto factor nodes. Undirected edges are defined between each factor node and its variables. Fig. 7 shows an example factor graph for a multivariate algebraic function along with its solution via summary product, as used in Sigma.

Given evidence about a subset of the variables – as stored in working memory factor nodes – messages are passed along the links and processed at the nodes to yield new messages. Each message along a link provides information about the values of the link's variable. Incoming messages at variable nodes are combined via pointwise product to yield outgoing messages, but with each outgoing message omitting from its product the incoming message on its link. Similar pointwise products occur at factor

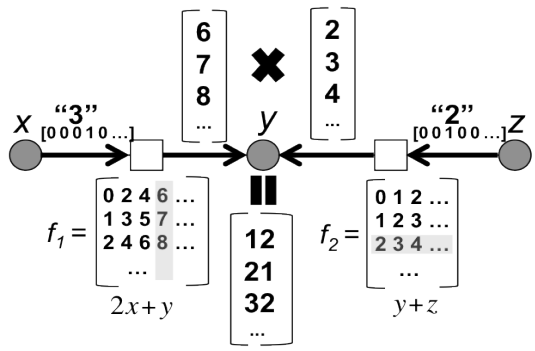

Fig. 7: Summary product computation over the factor graph for $f(x,y,z) = y^2+yz+2yx+2xz = (2x+y)(y+z) = f_1(x,y)f_2(y,z)$ of the marginal on y given evidence concerning x and z. Only the messages (and link directions) involved in computing y are shown.

nodes, but with the factor's function also included in the product; and then all variables not in the outgoing message are summarized out. Summarization typically occurs via *summation* – or *integration* for continuous functions – to yield marginals, or via *maximum* to yield the mode; however, maximum is also used in Sigma for marginals of universal variables. Message passing ends upon quiescence; i.e., when no new message is significantly different from the previous message along the same link.

Both conditions and actions can be *negated*, inverting the resulting function to yield $\not f = maximum(1\text{-}f, 0)$. True (1) becomes false (0) and vice versa. Intermediate values are similarly inverted, and functional values greater than 1 are treated as if they are 1 during the inversion. Fig. 8, for example, shows how an object can be removed from an image via a negated action that spans the entire plane for object 1.

```
CONDITIONAL Delete-1
   Actions: (Image – o:1 x:* y:*)
```

Fig. 8: Deletion of object 1.

Both conditions and actions also limit the direction in which messages are passed – those within condition subgraphs only move away from working memory while those within action subgraphs only move towards it. This provides the forward momentum central to procedural memory. *Condacts* – a neologism for *cond*itions and *act*ion*s* – provide the bidirectional message passing required for the full generality of factor graphs, as used for example in probabilistic reasoning, constraint satisfaction, signal processing, and (partial match in) declarative memory [7]. As condacts are not used in the results presented here, they aren't discussed further. We will also omit discussions of other aspects of Sigma not exploited here, such as learning.

Still, two remaining aspects of Sigma do require explication. The first, and the only one originally motivated by the needs of mental imagery, is the use of affine transformations; i.e., combinations of linear transformations with translations. Fig. 9 shows an example, where an affine transformation is used in the action of a conditional to scale a Z tetromino horizontally, in place. In general, a variable in a condition or an action may include a coefficient and an offset, where the coefficient must be a constant and the offset may be either a constant or a variable (although only constant offsets are used in the work described here). Affine transforms can be used in conditions, actions and condacts, but with a transformation in a condition (or the outgoing aspect of a condact) inverting what the same one does in an action (or the incoming aspect of a condact).

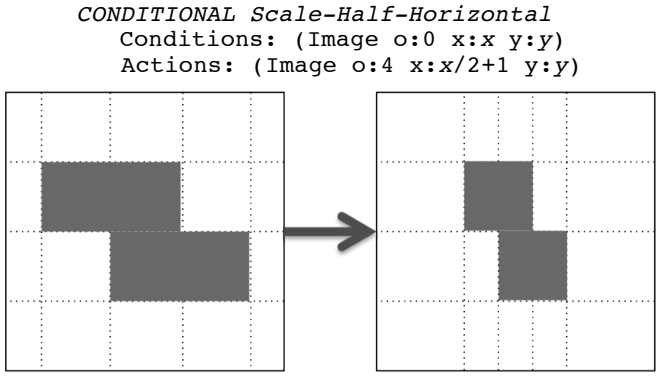

Fig. 9: Scaling a Z tetromino by half, horizontally, in place.

Although the affine transformation specified in Fig. 9's conditional may appear to involve just addition and multiplication of individual numbers, the figure makes it clear that such transformations actually operate on entire functions. In principle, affine transformation can and should be implemented by standard factor nodes that represent variants of *delta functions* [11]. However, delta functions are awkward and expensive to approximate via axially aligned slices, so specially optimized factor nodes that directly manipulate message slices, such as those in Fig. 9, are used instead. An offset shifts a whole piecewise linear function along a variable's dimension by modifying the dimension's slices, while a coefficient may, once again by modifying slices, expand, contract, or invert a dimension.

Once the slices have been modified, the resulting function may then need to be cropped and/or padded. Dimensions are not infinite in Sigma; each must be specified via minimum and maximum values, defining a domain that is closed at its dimensional minima and open at its corresponding maxima (this same half-open structure is also shared by regions). When a transformation extends a function beyond its dimensional bounds, it is cropped to fit back within these boundaries. When a transformation leaves areas within the boundaries undefined, the function is padded by assigning values to these areas. By default, closed-world predicates use a value of 0 and open-world predicates use a value of 1, corresponding for each to the standard value of *unknown*. Although originally motivated by mental imagery, affine transformations have since found important roles in Sigma across such areas as episodic memory, reflection, and reinforcement learning [11, 8].

The other aspect of Sigma used in the mental imagery results here is a capability for applying *piecewise linear filters* – generalizations of the constant tests typically found in rule conditions – to messages. A constant test is simply a filter that passes along only the portion of incoming messages matching the constant, via a filter that is 1 where the variable's domain equals the constant and 0 elsewhere. Sigma's filters can more generally specify arbitrary linear functions over regions. For example, in the condition `(Image o:o x:x y:[.01*y])`, the computation within the square brackets defines a filter that increases linearly with (the domain value) of `y`, with a slope of .01. The functional values in incoming messages are therefore pointwise multiplied by .01 times their `y` domain value. Such filters have been used, for example, in reinforcement learning to compute expected Q values via summarization (integration) over a weighted distribution of Q values [8]. They are leveraged in the next section in computing directional relationships among objects.

3 Results

The focus in this section is on key implications for mental imagery of the capabilities just described. This is not exhaustive, as new implications are continually being uncovered, but it does span the requirements mentioned in the introduction.

We can begin with the straightforward result that it is possible to translate, scale (shrink/enlarge), reflect, and rotate objects in images. Translation was covered in [11] and Fig. 9 demonstrated scaling in place, via a coefficient and an offset. Figs. 10 and 11 both start with the Z tetromino on the left of Fig. 9, with Fig. 10 then demonstrating reflection in place, via a negative coefficient and an offset, and Fig. 11 demonstrating rotation by 90° in place, via reflection and a swap of the *x* and *y* variables. As presently implemented, Sigma's affine transformations operate on individual variables (i.e., dimensions). By swapping variables – and reflecting when necessary – rotations by multiples of 90° are possible, as here, but not arbitrary-angle rotations. Two issues stand in the way: (1) Sigma's limitation to rectilinear, axially aligned, regions makes it complex and costly to represent the results of such rotations [11]; and (2) rotations at arbitrary angles require multivariate transforms. We are considering extending Sigma's function representation from orthotopic regions to (convex) *polytopic* regions – i.e., *n*D polygons – to allow representation of slices at arbitrary angles (as well as to enable more compact representations of complex objects). When this is in place, efficient multivariate transformation will be explored.

The conditionals in Figs. 9-11 demonstrate image composition – each adds one object (on its own plane) to the overall image – and Fig. 8 demonstrated object deletion. What hasn't been demonstrated is how the separate objects in an image can be combined into a single new plane, enabling hierarchies in which complex images can in turn be treated as objects in more complex images. Fig. 12 demonstrates this, with the object variable (*o*) from the condition – which ranges over the four planes in the image – being summarized out via *maximum* to yield a message to the action that is 1 wherever there is a 1 in any of the individual objects in the image.

The result of the processing in Fig. 12 is a new composite object that can be treated like any other object. For example, the left edge of this object – the slivers

```
CONDITIONAL Reflect-Horizontal
   Conditions: (Image o:0 x:x y:y)
   Actions: (Image o:5 x:4-x y:y)
```

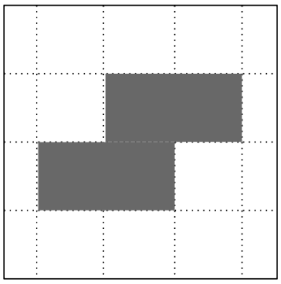

Fig. 10: Horizontal, in place, reflection of Z tetromino.

```
CONDITIONAL Rotate-90-Right
   Conditions: (Image o:0 x:x y:y)
   Actions: (Image o:3 x:4-y y:x)
```

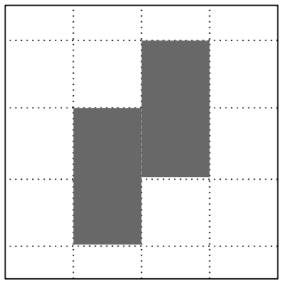

Fig. 11: 90° rotation of Z tetromino.

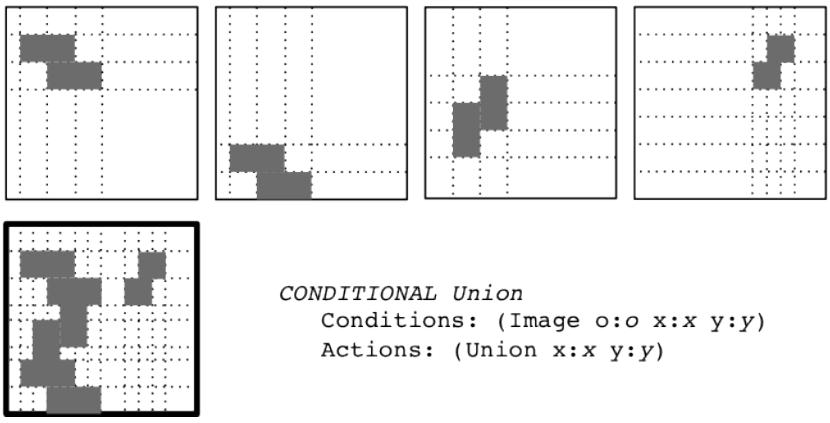

Fig. 12: Combining four object planes (top) into a single new plane (bottom left).

immediately to the right of blank areas – can be determined as in Fig. 13. This is an elementary perceptual operation that can extract useful information from images. Just as with the other imagery operations though, it occurs via a standard conditional that compiles down to a factor graph. In this instance, the conditional uses an offset in a negated condition to shift the image by ε (.0001 in this case) and then to invert it before multiplying by the original image. The result is 1s only for the sliver of the original image that is within ε to the right of a blank area. This approach turns out to perform edge detection without previously pixelating the image; instead, the thickness of the edge is a function of the offset.

```
CONDITIONAL Left-Edge
   Conditions: (Union x:x y:y)
               (Union – x:x-.0001 y:y)
   Actions: (Left-Edge x:x y:y)0
```

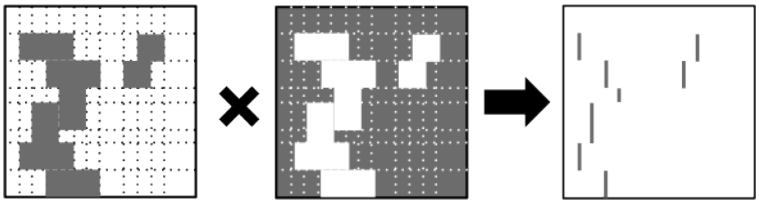

Fig. 13: Computation of left edge of composite object.

A second example of extracting useful information from a combination of objects was shown in Figs. 3 and 6, where conditions for separate image planes compute their overlap via the *product* aspect of summary product. It is then a simple step from there to a third example, where colliding pairs are detected via summarizing – by *maximum* – the two spatial dimensions in the vector of overlap planes (Fig. 14).

As a fourth and final example of information extraction from mental imagery, consider the problem of determining directional

```
CONDITIONAL Collision
   Conditions: (Overlap i:i x:x y:y)
   Actions: (Collision i:i value:true)
```

Fig. 14: Determine which objects collide.

information among objects, such as whether object 1 is to the right of object 2, or which object is topmost. Fig. 15 shows a conditional for the latter computation, where the topmost object is defined to be the one whose topmost point is above the topmost points of all of the other objects. It uses a filter in the condition to weight points in objects by their `y` (domain) values, decreasing as `y` increases. In generating a message for the action, by summarizing out `x` and `y` via *maximum*, this computes a function value for the object equal to the weight of its topmost point. The action then uses a unique variable in the `Topmost` predicate to select the most highly valued object; that is, the one whose topmost point is highest among all of the objects in the image.

```
CONDITIONAL Above
   Conditions: (Image o:o x:x y:[1-.1*y])
   Actions: (Topmost o:o)
```

Fig. 15: Determine which object is topmost.

Together these last four examples start to show how Sigma can extract useful information from the spatial interactions among objects in images, as the earlier examples show how to compose images from multiple objects, turn these composites into new objects, delete objects from images, and transform objects within images.

4 Conclusion

The mental imagery results presented here derive from a combination of: Sigma's core *n*D piecewise linear representation for functions/messages; its use of conditionals with conditions, actions and negations to define a factor graph; the generalization from constants to piecewise linear filters in conditionals; the addition of (optimized factor nodes for) affine transformations; and how the functions/messages are combined and reduced via the summary product algorithm. This combination enables the componential representation of continuous 2D images in terms of vectors of region-based objects; the addition and deletion of objects from these images; translation, scaling, reflection and (limited forms of) rotation of these objects; and the ability to extract implications from interactions among objects.

Although not a focus here, it is trivial via additional predicates to symbolically annotate these continuous objects. The initial step in extending this all from 2D to 3D imagery is also trivial, involving merely the addition of a `z` dimension. However, this hasn't yet been pursued because of the computational cost of processing these larger images. We are presently modifying Sigma's core representation so that slices need not span the entire space, and default-valued regions can be represented implicitly. These changes should reduce the size of the imagery functions and improve the efficiency of their processing. This should not only enable efficient exploration of 3D imagery, but also provide an important step in moving from orthotopes to polytopes (which should further simplify the representation of complex objects, while enabling exploration of arbitrary-angle rotations). We are also exploring the possibility of allowing more direct incorporation of Gaussians, or comparable functions, for more efficient representation of spatial, and other forms, of uncertainty.

Beyond these extensions, we need to look at incorporating these basic capabilities into naturalistic tasks that are tightly coupled with true perception; and, in the process,

evaluate whether this functionality is both sufficient and sufficiently efficient. Still, the results presented here do demonstrate a significant mental imagery capability that is built upon a set of more primitive mechanisms that are common to other cognitive capabilities within Sigma; for example, reinforcement learning [8] also leverages all of the capabilities listed at the beginning of this section (except for negation). It thus represents a significant step towards a functionally elegant grand unification.

Acknowledgments. This effort has been sponsored by the Air Force Office of Scientific Research and the U.S. Army. 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. Gunzelmann, G. Lyon, D.R.: Representations and processes of human spatial competence. Topics in Cognitive Science, 3, 741-759 (2011)
2. Cassimatis, N.: Polyscheme: A Cognitive Architecture for Integrating Multiple Representation and Inference Schemes. Ph.D. Dissertation. MIT Media Laboratory (2002)
3. Lathrop, S.D., Wintermute, S., Laird, J.E.: Exploring the functional advantages of spatial and visual cognition from an architectural perspective. Topics in Cognitive Science, 3, 796-818 (2011)
4. Trafton, J.G., Harrison, A.M.: Embodied spatial cognition. Topics in Cognitive Science, 3, 686-706 (2011)
5. Chandrasekaran, B., Banerjee, B., Kurup, U., Lele, O.: Augmenting cognitive architectures to support diagrammatic imagination. Topics in Cognitive Science, 3, 760-777 (2011)
6. Hutter, M.: Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability. Springer-Verlag, Berlin (2005)
7. Rosenbloom, P.S.: Combining Procedural and Declarative Knowledge in a Graphical Architecture. In: 10th International Conference on Cognitive Modeling (2010)
8. Rosenbloom, P.S.: Deconstructing reinforcement learning in Sigma. In: Fifth Conference on Artificial General Intelligence (2012)
9. Rosenbloom, P.S.: From memory to problem solving: Mechanism reuse in a graphical cognitive architecture. In: Fourth Conference on Artificial General Intelligence (2011)
10. Chen, J., Demski, A., Han, T., Morency, L-P., Pynadath, P., Rafidi, N., Rosenbloom, P.S.: Fusing symbolic and decision-theoretic problem solving + perception in a graphical cognitive architecture. In: Second International Conference on Biologically Inspired Cognitive Architectures (2011)
11. Rosenbloom, P.S.: Mental imagery in a graphical cognitive architecture. In: Second International Conference on Biologically Inspired Cognitive Architectures (2011)
12. Kschischang, F.R., Frey, B. J., Loeliger, H.: Factor Graphs and the Sum-Product Algorithm. IEEE Transactions on Information Theory, 47, 498-519 (2001)
13. Rosenbloom, P.S.: Bridging dichotomies in cognitive architectures for virtual humans. In: AAAI Fall Symposium on Advances in Cognitive Systems (2011)
14. Rosenbloom, P.S.: Implementing first-order variables in a graphical cognitive architecture. In: First International Conference on Biologically Inspired Cognitive Architectures (2010)
15. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press, Cambridge (2009)