*** To Appear in AISBQ, 2013 ****

The Sigma Cognitive Architecture and System

Paul S. Rosenbloom
University of Southern California

Sigma (Σ) is a nascent *cognitive system* – an integrated computational model of intelligent behavior, whether natural and/or artificial – that is based on a novel *cognitive architecture*: a model of the fixed structure underlying a cognitive system [1]. The core idea behind Sigma is to leverage *graphical models* [2, 3] – with their ability to yield state-of-the-art algorithms across the processing of signals, probabilities and symbols from a single representation and inference algorithm – in constructing a cognitive architecture/system that meets three general desiderata: *grand unification*, *functional elegance* and *sufficient efficiency*.

A *unified cognitive architecture* traditionally attempts to integrate together the complementary cognitive capabilities required for human(-level) intelligent behavior, with appropriate sharing of knowledge, skills and uncertainty among them. A *grand unified architecture* goes beyond this, in analogy to a grand unified theory in physics, to include the crucial pieces missing from a purely cognitive theory, such as perception, motor control, and emotion. This shifts issues of *embodiment*, *grounding* and *interaction* into the foreground, to converge with work on robot and virtual human architectures, but without then relegating traditional cognitive concerns to the background. Sigma approaches grand unification via a *hybrid* (discrete + continuous) *mixed* (symbolic + probabilistic) combination of: (1) graphical models, in particular *factor graphs* with the *summary product algorithm* [4]; plus (2) *piecewise-continuous functions* [5].

Functional elegance implies combining broad functionality – grand unification in this case – with simplicity and theoretical elegance. The goal is something like a set of *cognitive Newton's laws* that yield the required diversity of behavior from interactions among a small set of general primitives. If the primitives are combinable in a flexible enough manner, new capabilities continue to flower without the need for new modules; and integration occurs naturally through shared primitives. The Soar architecture, many of whose strengths and weaknesses have inspired choices in Sigma, took a similar approach in its early years [6], while AIXI can be seen as striving for an extreme version of it [7]. Although doubts remain as to whether natural cognitive systems are elegant in this manner – as opposed to mere evolutionary patchworks – and whether such elegance is even computationally feasible in artificial cognitive systems, developments such as rational analysis on the natural side [8] and graphical models on the artificial side provide continued promise. Despite the questions, functional elegance maintains its allure because, if it is in fact achievable, it should yield deeper and more elegant theories with broader scientific reach [9] that are ultimately easier to understand and apply.

Sufficient efficiency implies cognitive systems that execute quickly enough to support their anticipated uses. On the artificial side, the primary issue is speed of execution, but joined at times with boundedness. Graphical models potentially play a key role here, as they not only yield broad functionality in an elegant manner, but also state-of-the-art performance across this breadth. On the natural side, the primary issue is whether behavior is modeled at appropriate human time scales, independent of how much real time is required. Yet speed is also an important secondary consideration here, particularly as experiments and models scale up.

The remainder of this article summarizes progress to date in achieving a functionally elegant grand unification in Sigma. First, the currently implemented architecture/system is described at a high level. Then results are summarized across memory and learning, perception and mental imagery, decisions and problem solving, multiagent systems and theory of mind, and natural language. Sufficient efficiency is not a major focus in these results, other than indirectly through the pervasive use of graphical models; although significant progress has been made on aspects of it [10], more is required before Sigma will be ready to support complex real-time virtual humans and robots.

Sigma

The term cognitive architecture derives from an analogy with computer architecture, the fixed structure of a computer that provides a programmable system (that is, a *machine language*). In a cognitive architecture the concern is with the fixed structure that provides a (machine) language for expressing the knowledge and skills that comprise the learnable content of the cognitive system. But a computer system isn't just an architecture plus content, and nor necessarily is a cognitive system. Sigma is presently composed of three main layers: (1) the cognitive architecture; (2) knowledge and skills included on top of the cognitive architecture; and (3) the analogue of a *firmware* (or *microcode*) *architecture* that sits beneath the cognitive architecture. The cognitive architecture provides a language of *predicates* and *conditionals* that blend ideas from rule-based systems and probabilistic networks. It directly supports the layer of knowledge and skills on top of it. A firmware architecture traditionally provides a programmable level in between what is directly supported by hardware and what is desired in the computer architecture. Sigma's firmware architecture bridges its underlying implementation language (Lisp) and its cognitive architecture via a language of factor graphs and piecewise continuous functions (into which predicates and conditionals are compiled for execution). In this section, we first explore Sigma's firmware architecture and then its cognitive architecture. Its present knowledge and skills are implicit in the results discussed in the next section.

Factor graphs, in common with other forms of graphical models – such as Bayesian and Markov networks, and Markov and conditional random fields – provide an efficient means of computing with complex multivariate functions by decomposing them into products of simpler functions and then translating them into graphs for solution. From such graphs, the marginals of the individual variables – i.e., the function's values when all other variables are summarized out – can be computed efficiently, as can the function's global mode; for example, yielding maximum a posterior probability (MAP) estimation. Factor graphs in particular are undirected bipartite graphs that combine *variable nodes* for the variables in the function with *factor nodes* for the factors into which the function decomposes (Figure 1). Each factor node embodies a function, and connects to all variable nodes used in the function. Each variable node connects to the factor nodes that use it. Unlike Bayesian networks, factor graphs can be applied to arbitrary multivariate functions, not just to probabilistic ones.

The representation used for factor functions in the graph is a critical determinant of the expressibility of the resulting system. Sigma supports a hybrid mixed approach via a core representation based on piecewise continuous functions, which at present are limited to piecewise linear. The domain of each factor function is the cross product of its variables, implying an *n*-dimensional function when there are *n* variables. The overall function is specified in a piecewise linear manner over an array of rectilinear regions (Figure 2). This representation is general enough to approximate arbitrary continuous functions as closely as desired. Furthermore, restrictions on these functions – for example, to unit intervals with constant functions – can yield both discrete and symbolic functions. Functions can also be hybrids if they comprise multiple variables of different types.

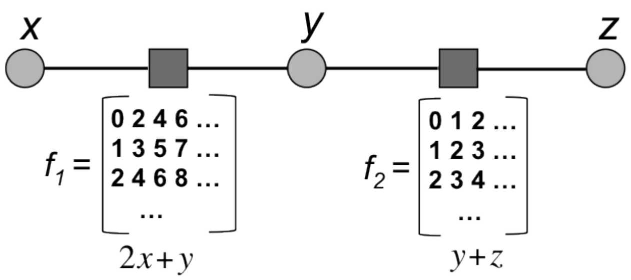

Figure 1: Factor graph for the algebraic function $f(x,y,z) = y^2+yz+2yx+2xz = (2x+y)(y+z) = f_1(x,y)f_2(y,z)$.

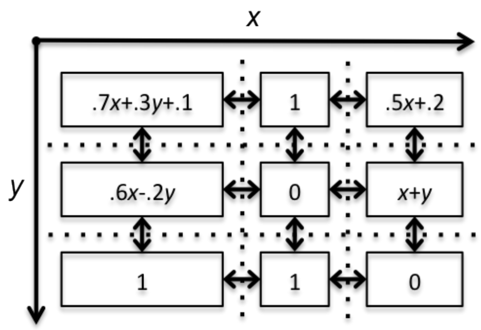

Figure 2: Bivariate function as a 2D array of regions with linear functions.

The processing cycle in Sigma's firmware architecture consists of a *graph-solution phase* followed by a *graph-modification phase*. Solving a factor graph requires applying one of the many inference algorithms available for computing the values of variables in graphical models. Such a solution typically involves providing *evidence* for some of the variables – for example, by fixing their values via functions in peripheral factor nodes – and then either computing the marginal distributions over the other variables individually or the modal value jointly over all of them. A message passing approach based on the summary product algorithm is used in Sigma to compute both marginals and modes (Figure 3). Messages are sent in both directions along links, from variable nodes to neighboring factor nodes and from factor nodes to neighboring variable nodes. This overall representation and processing is supported in Sigma's firmware architecture via four memories, for factor nodes, variable nodes, links, and messages (caching the last message sent in each direction along each link).

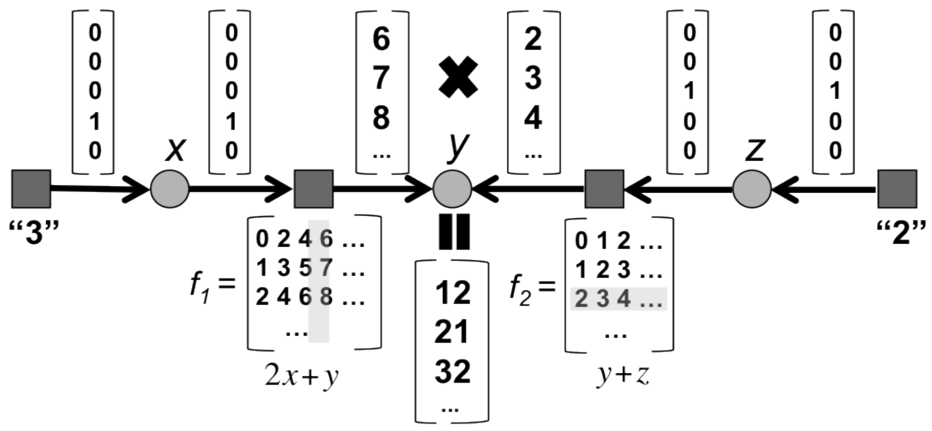

Figure 3: Summary product computation over the algebraic function in figure 1 of the marginal on *y* given evidence for *x* and *z*.

A message along a link always represents a distribution over the variable node's variables irrespective of its direction. When such a message is received at a variable node a new outgoing message is generated along each of its other links as the pointwise product of the incoming messages. This is the *product* aspect of the summary product algorithm. If the node is a factor node, the same pointwise product is computed, but included in the product is also the node's own function. Unlike at variable nodes, where the outgoing message is simply this product, further processing is required to compute the outgoing message here. Because the product includes all of the factor node's variables, not just those corresponding to the variable node on the outgoing link, all other variables must be *summarized* out before the message is sent. When computing marginals, Sigma uses *integration* for summarization. When computing modes it uses *maximum* instead.

The natural stopping criterion for the graph-solution phase – and thus the trigger for the start of the graph-modification phase – is *quiescence*; that is, when no significantly different messages remain to be sent. Sigma's message memory is modified dynamically during the graph-solution phase, but the graph-modification phase is ultimately responsible for altering the other three memories. At present, the graph-modification phase can alter functions maintained within factor nodes, in support of updating the cognitive architecture's working memory and some forms of learning, but it does not yet yield structural learning. *Working memory modification* and *gradient-descent learning* both modify factor functions in Sigma's graphs based on messages arriving at the factor nodes. The latter was inspired by work on local learning in Bayesian networks that showed results similar to backpropagation in neural networks, but with no need for an additional backpropagation mechanism [11]. *Episodic learning*, in contrast, updates temporally organized factor functions in Sigma based on changes over time in corresponding working-memory factor functions.

At the center of Sigma's cognitive architecture are two memories, *working memory* and *long-term memory*, each of which grounds out in the four firmware memories. The core of Sigma's *cognitive cycle* consists, à la Soar's, of accessing long-term memory until quiescence followed by decisions and learning, but with a generalized notion of what can be in long-term memory. Memory access is implemented by the graph-solution phase within the firmware cycle, while decisions and learning map onto the graph-modification phase. In addition, Sigma's cognitive cycle includes a perceptual phase prior to the graph-solution phase and a motor phase after the graph-modification phase. Sigma's cognitive cycle is intended to map onto the 50 msec cycle found in humans and many other cognitive architectures [10].

Working memory in Sigma is based on predicates, while long-term memory is based on conditionals. A predicate specifies a class of piecewise continuous functions via a name and a set of typed arguments – such as `Board(x:dimension y:dimension tile:tile)` for an Eight Puzzle board with continuous *x* and *y* dimensions and a discrete *tile* dimension – providing a cognitive data structure for storage of short-term information. Working memory embodies the evidence that drives processing in the long-term memory graph. Predicates can be either *closed world* or *open world*, depending on what is assumed when initializing working memory about values not in evidence. Predicates can also be mixed and/or hybrid, and in combination can enable richly structured representations [5].

A conditional in long-term memory specifies a knowledge fragment in terms of *predicate patterns* plus an optional *conditional function* (Figure 4). A pattern includes the predicate's name plus a constant or a variable for each specified argument. In the firmware architecture, a constant is matched to a message by a factor node containing a piecewise-constant function that is 1 in regions corresponding to the constant and 0 everywhere else. It took some time to realize, but was obvious in retrospect, that such a constant test is merely one special case of a general *piecewise-linear filter* in which each region may specify an arbitrary linear function, and that the firmware architecture already supports the full generality of such filters. The conditional language has therefore also been generalized to support the use of such filters in patterns. A second generalization has likewise been introduced for variables in support of *affine transforms*; that is, combinations of linear transforms and translations that together can yield object translation, rotation, scaling and reflection. These transforms are central to work on mental imagery in Sigma [12, 13], as well as playing significant roles in other capabilities of interest, such as episodic memory, reflection, and reinforcement learning. In essence, all numeric variables in Sigma – whether discrete or continuous and whether visual, temporal or other – are fragments of mental images to which affine transforms can be applied.

```
CONDITIONAL Transition
   Conditions: Location(state:s x:x)
                Selected(state:s operator:o)
   Condacts: Location*Next(state:s x:nx)
   Function: .2<Right(0)=0> .8<Right(0)=1>
             .2<Right(1)=1> .8<Right(1)=2>
                              …
             .8<Left(5)=4> .2<Left(5)=5>
```

Figure 4: Example conditional for a probabilistic action model (or transition function) in a 1D grid task in which the actions don't always behave as requested.

When used in conditionals, predicate patterns can function as *conditions*, *actions*, and *condacts*. Conditions and actions are akin to the like-named patterns in rules, and their functionality is comparable. Conditions match to evidence in working memory, passing on the successful results for further use. Actions propose changes to working memory. Condacts, a neologism for *cond*itions and *act*ion*s*, fuse the effects of these two, both matching to working memory and suggesting changes to it. They combine local constraint from the predicate's own portion of working memory with global constraint from the rest of memory to support, for example, partial matching in declarative memory, constraint satisfaction, signal processing, and general probabilistic reasoning.

Conditionals compile down to factor graphs in the firmware layer in a manner that is inspired by how the Rete match algorithm [14] handles conditions in rules, but extended to handle actions and condacts. The main difference between conditions and actions versus condacts is that messages pass in only one direction for the former two – away from working memory for conditions and towards it for actions – while messages pass bidirectionally for the latter. Conditional functions are also linked to this graph, extending the basic Rete idea for them as well. Although the term *conditional* is intended to evoke the conditionality found in both rules and (conditional) probability distributions, this should not be taken to imply that rules are the only structural form of knowledge available, nor that conditional probabilities are the only functions representable via conditional functions. The blending enabled by the firmware architecture is at a deep enough level and a small enough granularity that a substantially larger space of possibilities emerges.

Decisions in Sigma, in the classical sense of choosing one among the best operators to execute next, are mediated through the introduction of an architecturally defined *selection predicate*. Operator decisions occur just as do selections of new working memory values for any other predicates, except that Soar-like impasses may occur during operator selection. An impasse occurs when there is insufficient knowledge available for making a decision, such as when there are no eligible operators for selection, or there are multiple candidates and insufficient knowledge to select among them. Impasses lead to reflective processing across a hierarchy of metalevel states, where the goal is to resolve the corresponding impasses by providing knowledge that, for example, determines which operator to select.

Implementation of multiagent systems in Sigma involves the addition of an *agent* argument to the selection and impasse predicates, and to any user-defined predicate whose contents can vary by agent. This enables decisions and impasses to occur on an agent-by-agent basis, but with sharing of knowledge structures across them when appropriate.

Results to Date

The results generated so far via Sigma span memory and learning, perception and mental imagery, decisions and problem solving, multiagent systems and theory of mind, and natural language. Memory results include demonstrating how both procedural and declarative memories can be defined idiomatically via conditionals and predicates [15]. A rule-based procedural memory is based on conditions and actions over closed-world predicates. Declarative memory is based on condacts over open-world predicates plus functions. Both semantic memory and episodic memory can in this way support retrieval from long-term memory based on partial matches to evidence in working memory. Semantic memory is based on a Bayesian classifier that retrieves/predicts both object categories and features not in evidence via marginals that are computed from learned regularities over many examples. Episodic memory stores a temporally organized history of working memory, enabling the best matching past episode to be retrieved as a distinct individual via MAP.

All of the learning results to date involve modifying conditional functions. Episodic learning maintains the history of changes to working-memory predicates in functions specified within automatically generated episodic conditionals. Gradient-descent learning modifies conditional functions based on messages that arrive at their factor nodes. With gradient-descent learning over appropriate conditionals in Sigma, it is has been possible to demonstrate forms of supervised learning, unsupervised learning – in a manner akin to expectation maximization – learning of action models (i.e., transition functions) and maps (relating perceived objects to their locations), and reinforcement learning (RL) [16, 17]. Supervised, unsupervised and map learning, plus model-based RL, all proved possible with no other change to the architecture than the addition of gradient-descent learning. However, to support both model-free RL and the learning of action models, an additional enhancement to the architecture was required to make pairs of successive states available for learning within single cognitive cycles.

Perception has been demonstrated in Sigma via a conditional random field (CRF) that computes distributions over perceived objects from noisy feature data, and via a localization graph that yields distributions over (current and past) locations from distributions over (present and past) objects and a map [18]. These two independent graphs can be combined into a single larger graph that yields distributions over locations based on noisy feature information.

Mental imagery leverages conditionals along with piecewise-linear functions that can be continuous, discrete or hybrid depending on the kind of imagery involved [12, 13]. As described earlier, the Eight Puzzle board can be represented, for example, as a 3D hybrid function, with continuous *x* and *y* dimensions plus a discrete *tile* dimension. Results in mental imagery have spanned object composition and deletion; object translation, scaling, inversion, and rotation (at multiples of 90°); and extraction of features from composite objects, such as overlaps, edges, and directional relationships.

Decision making and problem solving have been demonstrated in a Soar-like manner, with preferences encoded via functional values that combine to determine what operator is chosen on each cycle [19], and impasses occurring when decisions cannot be made. Problem solving can occur either via a sequence of steps within the base level, or across meta-levels as impasses occur. Un-Soar-like decision-theoretic decision making has also been demonstrated, with a multi-step POMDP implemented via conditionals that generate preferences for operator selection based on probabilistic projection [18]. Such a POMDP has been combined with the joint perception+localization graph described above to yield a single system in which object perception feeds localization, and localization feeds decision making, all within a single decision

[18]. Initial work on theory of mind in Sigma has built on its multiagent capabilities plus POMDPs to demonstrate both the derivation of Nash equilibria for two-person, one-shot games, and intertwined multistep, multiagent POMDPs.

Early work on natural language (NL) has demonstrated a form of statistical response selection that is modeled after (part of the) approach taken in the NPCEditor [20]. Given the words in an input sentence and appropriate statistical background knowledge, a choice is made of an output sentence from a set of prespecified candidates. We have also scaled up semantic memory and learning in support of particular NL classification tasks, such as word sense disambiguation and part of speech tagging [16].

Conclusion

Although Sigma is still in a fairly early stage of development, and is not yet ready for large-scale real-time tasks, demonstrations to date indicate some of what is possible when graphical models are at the heart of a cognitive architecture/system. The beginnings of grand unification have been demonstrated via hybrid representations, and via combinations of perception and imagery with cognitive decision making and problem solving. Functional elegance has been demonstrated via a range of memory, learning, and decision making capabilities supported on a uniform base. The demonstration via factor graphs of state-of-the-art algorithms such as Rete for rule match and conditional random fields for vision also foreshadows sufficient efficiency. Much more of course remains to be done, but the path and its promise should be evident.

Acknowledgement

This effort has been sponsored by the U.S. Army, the Air Force Office of Scientific Research and the Office of Naval Research. 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. The work on Sigma described here has involved Junda Chen, Abram Demski, Teawon Han, Anton Leuski, Stacy Marsella, Louis-Philippe Morency, David Pynadath, Nicole Rafidi, Sanjay Raveendran, Kenji Sagae and Volkan Ustun.

References

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

2. Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA: Morgan Kaufman.

3. Koller, D. & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA: MIT Press.

4. Kschischang, F. R., Frey, B. J. & Loeliger, H. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, *47*, 498-519.

5. Rosenbloom, P.S. (2011). Bridging dichotomies in cognitive architectures for virtual humans. *Proceedings of the AAAI Fall Symposium on Advances in Cognitive Systems*.

6. Laird, J. E., Newell, A. & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence, 33*, 1-64.

7. Hutter, M. (2005). *Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability*. Berlin: Springer-Verlag.

8. Anderson, J. R. (1990). *The Adaptive Character of Thought*. Mahwah, NJ: Lawrence Erlbaum Associates.

9. Deutsch, D. (2011). *The Beginning of Infinity: Explanations that Transform the World*. London, UK: Penguin Books.

10. Rosenbloom, P. S. (2012). Towards a 50 msec cognitive cycle in a graphical architecture. *Proceedings of the 11th International Conference on Cognitive Modeling*.

11. Russell, S., Binder, J., Koller, D. Kanazawa, K. (1995). Local learning in probabilistic networks with hidden variables. *Proceedings of the 14th International Joint Conference on AI.*

12. Rosenbloom, P. S. (2011). Mental imagery in a graphical cognitive architecture. *Proceedings of the 2nd International Conference on Biologically Inspired Cognitive Architectures* (pp. 314-323). Arlington, VA: IOS Press.

13. Rosenbloom, P. S. (2012). Extending mental imagery in Sigma. *Proceedings of the 5th Conference on Artificial General Intelligence* (pp. 272-281). Oxford, UK: Springer.

14. Forgy, C. L. (1982). Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence*, *19*, 17-37.

15. Rosenbloom, P. S. (2010). Combining procedural and declarative knowledge in a graphical architecture. In D. D. Salvucci & G. Gunzelmann (Eds.), *Proceedings of the 10th International Conference on Cognitive Modeling* (pp. 205-210).

16. Rosenbloom, P. S., Demski, A., Han, T. & Ustun, V. (in preparation). Gradient-descent learning in the Sigma cognitive architecture/system.

17. Rosenbloom, P. S. (2012). Deconstructing reinforcement learning in Sigma. *Proceedings of the 5th Conference on Artificial General Intelligence* (pp. 262-271). Oxford, UK: Springer.

18. Chen, J., Demski, A., Han, T., Morency, L-P., Pynadath, D., Rafidi, N. & Rosenbloom, P. S. (2011). Fusing symbolic and decision-theoretic problem solving + perception in a graphical

cognitive architecture. *Proceedings of the 2nd International Conference on Biologically Inspired Cognitive Architectures* (pp. 64-72). Arlington, VA: IOS Press.

19. Rosenbloom, P. S. (2011). From memory to problem solving: Mechanism reuse in a graphical cognitive architecture. *Proceedings of the 4th Conference on Artificial General Intelligence* (pp. 143-152). Berlin: Springer.

20. Leuski, A., & Traum, D. (2010). NPCEditor: A tool for building question-answering characters. *Proceedings of The 7th International Conference on Language Resources and Evaluation*.