To appear in *Proceedings of the 36th Annual Conference of the Cognitive Science Society*, 2014.

Deconstructing Episodic Memory and Learning in Sigma

Paul S. Rosenbloom (Rosenbloom@USC.Edu)
Institute for Creative Technologies & Department of Computer Science, University of Southern California
12015 Waterfront Dr.
Playa Vista, CA 90094 USA

Abstract

In an experiment in *functional elegance*, episodic memory and learning have been deconstructed in the Sigma cognitive architecture in terms of pre-existing memory and learning mechanisms plus a template-based structure generator. As a side effect, base-level activation also becomes deconstructed in terms of a learned temporal prior.

Keywords: Cognitive architecture; episodic memory; learning; base-level activation.

Episodic memory is a core competency in human cognition (Tulving, 1983), but not yet a pervasive capability in cognitive architectures. It is relevant in both modeling human cognition and in creating artificial intelligence. Its core functionality includes the ability to store the history of what has been experienced (autobiographical/temporal) and to retrieve and reuse information from relevant past episodes given appropriate cues. It may also support replay of fragments of history given a starting location in it.

One of the earliest implementations of episodic memory was in the Basic Agent (Vere & Bickford, 1990). More recently, episodic memories have been added to Soar (Nuxoll & Laird, 2012) and Icarus (Stracuzzi *et al*., 2009). ACT-R's declarative memory is also relevant (Anderson *et al.*, 2004); however, as with most forms of instance-based learning, it does not explicitly represent time or adjacency (but see Altmann & Gray, 1998). Episodic memory also relates to case-based reasoning (Kolodner, 1993), but it is task independent and not solution oriented.

Sigma (Rosenbloom, 2013) – a new cognitive architecture built around the state-of-the-art generality and efficiency of *graphical models* (Koller & Friedman, 2009) – also embodies an episodic memory, inspired primarily by Soar, but also making contact with ACT-R. The initial focus was on episodic storage and retrieval via the same memory structures – *conditionals* – that also yield a concept-based semantic memory and a rule-based procedural memory (Rosenbloom, 2010). Conditionals provide a deep blending of the conditionality found in both rule-based systems and probabilistic networks, all grounded in message passing – via a variant of the *summary-product algorithm* – in factor graphs (Kschischang, Frey & Loeliger, 2001).

This early work on memory illustrated, and was driven by, a key desideratum in Sigma's overall development – *functional elegance* – which focuses on architectures that are broadly capable yet simple and theoretically elegant. In other words, the goal is to generate the wealth of requisite functionality through the interactions among a small set of very general mechanisms, and thus to yield a deeper theory with broader explanatory reach (Deutsch, 2012).

More recently, functional elegance has been guiding the addition of learning to Sigma, with the most significant piece being a gradient-descent mechanism for learning the parameters in factor nodes (Rosenbloom *et al.*, 2013). Given appropriate conditionals, this has proven sufficient for both supervised and unsupervised classifier learning, the acquisition of maps (in SLAM), and reinforcement learning (along with the acquisition of action models).

The work reported here combines the earlier ideas for functionally elegant representation with the newer ideas for functionally elegant learning, plus one additional idea – *automatic template-driven structure creation* – to yield automated episodic learning and retrieval. Episodic learning was actually one of the earliest forms implemented in Sigma, but via an isolated special purpose module. The current work shows how such learning can instead be deconstructed in terms of a combination of more general components that already exist in Sigma plus a template-driven structure generator for episodic memory. The contributions of this work concern these architectural results rather than specific matches to human data.

This work also demonstrates how *base-level activation*, as pioneered in ACT-R (Anderson *et al.*, 2004) and later reimplemented in Soar, can be deconstructed in terms of a learned *temporal prior* that forms the backbone of episodic memory (biasing retrieval towards more recently learned episodes, as well as ones that have been more recently and frequently accessed). Rather than a planned part of the architecture, it was a true discovery that gradient-descent learning mimics base-level activation in episodic memory. Several additional, albeit smaller, such discoveries are also covered in the body of this article.

These two deconstructions – with a particular focus on episodic learning – are at the heart of the contribution of this paper. They reveal how such capabilities can be supported in a relatively uniform non-modular manner within a cognitive architecture. This in turn yields a deeper understanding of these capabilities in an architectural context, and more generally of how to approach the development of functionally elegant architectures.

Sigma

Sigma is composed of two layers: the *cognitive architecture* plus a *graphical architecture* beneath it. The cognitive architecture is based on *predicates* and *conditionals*. A predicate represents a relation – such as the number of legs

– via a name and a set of typed arguments. Each predicate is allocated a distinct segment of the cognitive architecture's working memory that encodes the predicate's status in terms of a function over its arguments.

The types used in predicate arguments may be discrete – symbolic or numeric – or continuous, and may vary in size. For example, a concept label can be represented as a predicate with one symbolic argument: `Concept(value:{walker table dog human})`. Or, the state of the board in the Eight Puzzle can be represented as a predicate with one discrete argument (for the *tile*) and two continuous arguments (for *x* and *y*): `Board(x,y:[0-3) tile:[0:8])`. At base, all types are continuous in Sigma, but discrete types fragment the number line into unit-length regions, and symbolic types associate symbols with these regions. Similarly, all functions over predicates are at base piecewise linear (Figure 1), enabling approximation of arbitrary continuous functions, but with discrete functions reducing to piecewise constant. Symbolic function ranges are restricted to {0, 1}, for {false, true}.

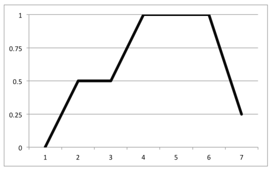

Figure 1: 1D piecewise-linear function.

A conditional represents a fragment of generalized conditional knowledge via a set of predicate patterns plus an optional function over variables in the patterns (Figure 2). Predicate patterns may act as *conditions* or *actions*, as in standard rules; however, they may also act as *condacts* – a composite bidirectional form that supports the conditionality found in probabilistic networks when conjoined with functions for the distributions. Conditionals are the basis for Sigma's long-term memory, with gradient descent modifying the functions within them. *Parameter tying* – a technique from HMMs – enables multiple conditionals to share, and even learn, the same function (Figure 3).

```
CONDITIONAL Legs-Time*Learn
  Conditions: Time(value:t)
              Legs(value:l)
  Function(t,l): <…>
```

Figure 2: Conditional for learning *Legs* given *Time*.

```
CONDITIONAL Legs-Time*Select
  Conditions: Legs(value:l)
  Condacts: Time*Episodic(value:t)
  Function(t,l): Legs-Time*Learn
```

Figure 3: Conditional for rating previous episodes based on number of *Legs* (with parameter tying).

Processing in Sigma proceeds through a sequence of cognitive/decision cycles; each of which involves accessing long-term memory and then deciding what changes to make in working memory – including selection of operators to be applied in problem solving – and adjustment (via gradient descent) of the values in conditional functions. Time is represented discretely, and incremented once per cycle. Its value is accessible via a temporal predicate with a single discrete argument: `Time(value: [0:999998])`.

Sigma's graphical architecture supports *factor graphs*; i.e., undirected graphical models constructed from variable and factor nodes. Factor nodes encode functions, and are linked to the variable nodes with which they share variables. A factor graph implicitly represents the function defined by multiplying together the functions in its factor nodes. Or, equivalently, a factor graph decomposes a single complex multivariate function into a product of simpler factors.

The *summary-product algorithm* computes messages at nodes and passes them along links. An output along a link from a variable node is the product of the inputs along its other links. An output along a link from a factor node is the product of the node's function times the inputs along its other links, with the variables not in the target variable node then summarized out, either by *integrating* them out to yield marginals or *maximizing* them out to yield maximum a posteriori (MAP) estimates.

The cognitive architecture's working memory compiles into a subregion of the graphical architecture's factor graph, with its contents represented via factor functions. Conditionals compile into more complex graphs, with their functions stored in their own factor nodes. Memory access in the cognitive architecture maps onto message passing in the induced factor graph, with gradient descent occurring locally at the relevant factor nodes based on the messages they receive (Russell *et al*., 1995; Rosenbloom *et al*., 2013).

Episodic Memory

In cognition, episodic memory is generally considered to be one part of a more comprehensive *declarative memory* that stores facts of all sorts. However, there is less of a consensus concerning whether declarative memory is a single uniform structure, as in ACT-R, or whether there are distinct modules for past history (episodic memory) versus world knowledge (semantic memory), as in Soar. Sigma occupies a middle ground, with all declarative memories structured as *classifiers* within a common factor graph, and with the same architectural processes operating on these classifiers, but with some details of the classifiers varying across types of knowledge (Rosenbloom, 2010).

Sigma's declarative memories are currently *naïve Bayes classifiers* – combining prior probabilities of concepts multiplicatively with conditional probabilities of features given concepts – that support the three central processes of declarative memory: (1) *learning* a new fact (moving it into long-term memory); (2) *selecting* an old fact (choosing which is best to retrieve); and (3) *retrieving* the selected fact (moving its aspects into working memory). For episodic memory, this maps onto: (1) learning a new episode, by storing into long-term memory an association between the current time and the features of the current situation; (2) selecting an old episode, by choosing the "best" previous episode given the current situation; and (3) retrieving an old episode, by accessing the features associated with it.

As an example, first consider Sigma's semantic memory, which is simpler in detail than its episodic memory. Semantic memory represents concepts and their attributes via a prior distribution over the concept plus conditional distributions over the attributes given the concept (Figure 4). A simple structure is sufficient for this in long-term memory, with one conditional per attribute (as in Figure 5) plus one for the concept prior. Via the summary-product algorithm, messages from the attribute variables – each of which is based on the evidence for the attribute times the conditional distribution of the attribute given the concept – are multiplied together, along with a message from the concept's prior distribution, to yield a posterior distribution over the concept. Semantic selection is based on this distribution. Semantic retrieval is based on messages back to the attributes from the concept – leveraging the bidirectionality provided by condacts – to yield attribute predictions. Although retrieval here could in principle be based on just the attribute values associated with the selected concept, it actually leverages the full concept posterior to generate more accurate predictions. Semantic learning is driven by the messages arriving at the factor nodes that store the distributions. These messages implicitly define the local gradient at these nodes.

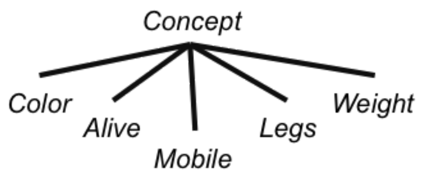

Figure 4: Semantic memory classifier.

```
CONDITIONAL Legs-Concept
  Condacts: Concept(value:c)
            Legs (value:l)
  Function(c,l): <…>
```

Figure 5: Conditional for *Legs* given *Concept*.

In contrast with semantic memory, time is at the heart of episodic memory. Episodes occur in the past, and their ages influence selection, typically via a function that tails off exponentially into the past. In Sigma, an exponential function over time is learned, rather than prespecified, via gradient descent over a conditional function (Figure 6). On each cycle, the current time is increased and then the whole function is normalized. An exponential results (Figure 7) because earlier times have been normalized more often.

```
CONDITIONAL Time*Learn
  Conditions: Time(value:t)
  Function(t): <…>
```

Figure 6: Conditional for temporal learning.

Although both semantic and episodic memory are naïve Bayes structures, episodic memory is instance-based rather than summative, with time replacing the concept and feature values at specific times replacing concept attributes (Figure 8). The detailed structure of episodic memory also turns out to be more complex due to the need to distinguish between

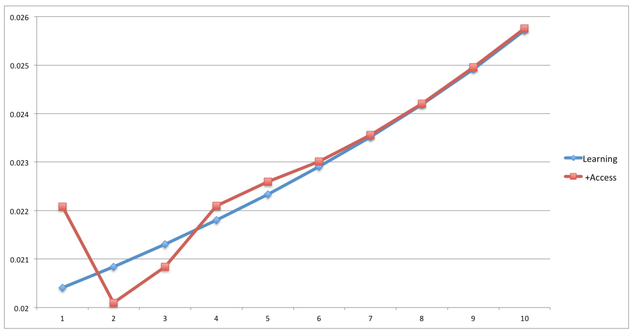

Figure 7: Learned exponential temporal function (and its modulation by episodic access).

the past and the present: episodic learning depends on what is true now; episodic selection depends on matching what is true now to what was true in the past; and episodic retrieval depends on what was true in the past.

In working memory this temporal distinction is instantiated via a pair of implicitly defined *working-memory buffers*, each of which comprises its own set of predicates. The current-state buffer contains the core predicates that represent the state of the problem to be solved, such as `board` for the Eight Puzzle, plus the architecturally generated `Time` predicate. The past-state – or *episodic* – buffer[1] has automatically generated predicates that mirror these – e.g., `Board*Episodic` mirrors `Board` – plus `Time*Episodic` for past times. The current-state buffer existed prior to this work, but the episodic buffer is new.

As with semantic memory, the early work on episodic memory and learning required only one conditional per attribute, plus one for the prior. However, in a full integration, the temporal distinction dictates mapping these three processes onto distinct conditionals that operate on the appropriate buffers.[2] For each feature, its trio of conditionals shares one function – via parameter tying – to link what is learned to what is selected and retrieved. A pair of conditionals for time – based on `Time` and `Time*Episodic` – also share a single function, with the former (Figure 6) used in learning the exponential temporal prior and the latter (Figure 9) used in accessing it during episodic selection and retrieval.

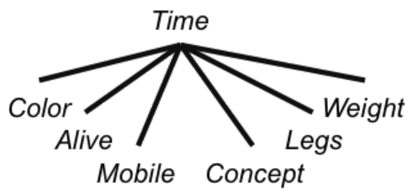

Figure 8: Episodic memory classifier.

[1] A concept related, but not identical, to the episodic buffer proposed in (Baddeley, 2000).

[2] At a level of detail beyond the scope of this paper, selection exploits a *next-state buffer*, previously developed for action modeling (Rosenbloom, 2012), rather than the current-state buffer.

```
CONDITIONAL Time*Access
  Condacts: Time*Episodic(value:t)
  Function(t): Time*Learn
```

Figure 9: Conditional for temporal access.

The `Time` predicate is automatically initialized when Sigma starts up. If episodic memory is enabled, template-based structure creation kicks in to initialize the `Time*Episodic` predicate and the two temporal conditionals. Each core state predicate also then becomes an episodic feature, leading to the template-driven creation of the corresponding episodic predicate plus the three conditionals that support its role in episodic learning, selection, and retrieval.

Episodic processing aligns with semantic processing, but again with more complexity. Episodic selection occurs via selection conditionals (Figure 3) that combine feature evidence from the current-state buffer with learned distributions for features given time to yield messages that rate episodic times. These messages are combined multiplicatively in the episodic buffer with each other, and with a message that encodes the temporal prior, to yield a full temporal posterior that is used in two distinct ways.

First, the temporal posterior supports selection of the best previous time, enabling episodic retrieval of its instance-based features via retrieval conditionals that consider both the episodic time and the learned distributions for past features given the time in retrieving episodic feature values (Figure 10). This contrasts with the summative retrieval in semantic memory, where the best attribute predictions are based on the full posterior concept distribution rather than just the single best concept. This difference is implemented by summarizing out the concept in semantic memory via *integration*, while using *maximum* in episodic memory. The choice of summarization operation can be per conditional, enabling local choice of whether to use the sum-product or max-product variants of the summary-product algorithm.

```
CONDITIONAL Legs-Time*Retrieve
  Conditions: Time*Episodic(value:t)
  Condacts: Legs*Episodic(value:l)
  Function(t,l): Legs-Time*Learn
```

Figure 10: Conditional for retrieving number of *Legs*.

Second, because messages pass in both directions between the learned temporal prior and the episodic buffer – due to the condact in Figure 9 – the temporal posterior also yields a message back to the stored temporal function, causing gradient descent to modify the temporal prior during access. As learning proceeds, the temporal prior thus not only reflects the primary exponential effects of learning from the current time, but it also exhibits secondary recency and frequency effects due to learning from the temporal posterior during access (Figure 7). These secondary effects contribute to mimic base-level activation, even though: (1) they occur via learning, and (2) the temporal posterior from which this learning occurs reflects the full distribution over the time, rather than just the selected time. This general approach also lends itself to potential incorporation of other factors into the temporal prior, taking it further and further from the simple exponential yielded by time of learning.

Beyond learning a distribution over past times, episodic learning also must acquire distributions over episodic features given time. This occurs via gradient descent at the function factor nodes in the learning conditionals (Figure 2), based on messages from the current-state buffer to them.

Results

To illustrate the behavior of episodic memory in Sigma, a simple artificial task has been implemented that uses the same features as earlier work in semantic memory: *Concept* in {walker, table, dog, human}, *Color* in {silver, brown, white}, *Alive* in {false, true}, *Mobile* in {false, true}, *Legs* discrete in [0,4], and *Weight* continuous in [0,500). The system first experiences, and learns from, the four full instances in Table 1. It then experiences the seven partial instances in Table 2. These latter serve as queries, although they are learned as well – there is no real difference between a learning situation and a retrieval situation in Sigma, since both activities occur every cycle. Table 2 also shows which prior episode is selected for each of these queries.

Table 1: Sequence of four full instances.

	Concept	*Color*	*Alive*	*Mobile*	*Legs*	*Wgt.*
T1	walker	silver	false	true	4	10
T2	human	white	true	true	2	150
T3	human	brown	true	true	2	125
T4	dog	silver	true	true	4	50

Table 2: Sequence of seven partially specified queries.

	Queries	Best
T5	*Concept*=walker	T1
T6	*Color*=silver	T4
T7	*Alive*=false, *Legs*=4	T1
T8	*Alive*=false, *Legs*=2	T3
T9	*Concept*=dog, *Color*=brown	T4
T10	*Concept*=walker, *Color*=silver, *Alive*=true	T1
T11	*Alive*=false	T8

Episode selection is based on the full temporal posteriors found in Figure 11. Each curve in the figure shows the posterior over the previous episodes for one query episode. Because one cycle is needed to learn an episode, the last point in each curve is for two episodes prior to it. For each query, the highest valued episode on its curve is selected.

Figure 7 showed the temporal prior for these eleven episodes, both when only updated as episodes are learned and when access also contributes. The most frequently retrieved episode is T1, which also yields the largest upwards deviation from an exponential. T4 is the next most frequent, and shows the next largest bump. The overall result is retrieval behavior that mimics base-level activation.

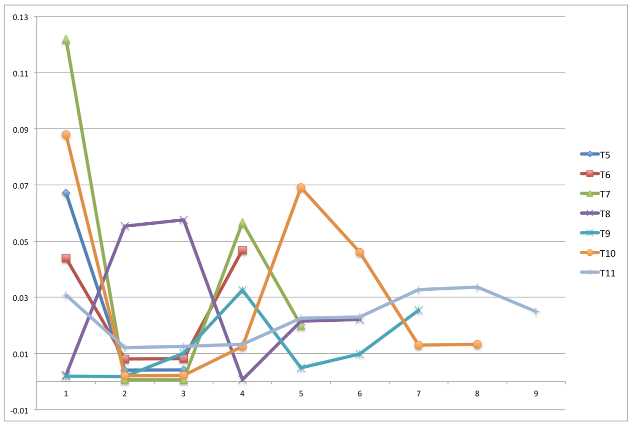

Figure 11: Temporal posterior for the seven queries.

After learning the four instances in Table 1, the six episodic functions, one per feature, record the histories of their features. For example, Table 3 displays the function for *Concept*. The correct values at each time step are learned with much higher ratings than the incorrect values. Episodes T2 and T3 are combined into a single column here due to Sigma's elimination of function-spanning boundaries when the corresponding pairs of regions across them have equivalent functions. This directly yields the episodic memory optimization of only recording feature changes.

Table 3: Episodic function learned for the *Concept* feature.

	T1	T2–T3	T4
walker	.85	.05	.05
table	.05	.05	.05
dog	.05	.05	.85
human	.05	.85	.05

Let's now examine the sequence of retrievals in Table 2, focusing first on times 5-7. At time 5, the only episode that matches the cue – T1 – is retrieved. At time 6, two episodes match – T1 and T4 – but the more recent one (T4) is retrieved. At time 7, two episodes match at least one of the cued features – T1 and T4 – but the retrieved episode (T1) matches both, and so is better despite being learned earlier.

These three initial retrievals demonstrate how selection occurs via partial match, preferring episodes that match cued features, with further preferences for matching multiple features and for more recent learning. When these latter preferences conflict, as at time 7, match quality trumps recency. All of this falls out of the multiplicative integration of feature matches and the temporal prior that is dictated by the naïve Bayes structure of the classifier.

At time 8, no stored episode matches both cues – the first matches T1 and the second matches T2 and T3 – requiring a more sophisticated form of partial match, where not all features in evidence need be matched by stored episodes. This is, however, still computed as before, via multiplicative integration across features. In this particular case, T3 dominates T2 because it is more recent while including the same cue. T3 also dominates T1, but for a less obvious reason. Given the details of the gradient descent algorithm, correct values for large argument domains yield higher ratings than do correct values for smaller domains; and here *Legs* has four possible values and *Alive* only two. Thus a complex tradeoff occurs, where episode T1 is preferred because of the match of *Alive* and a higher temporal prior – even though T1 occurred earlier, its prior is higher because it has been accessed repeatedly – but the preference yielded for T3 by the match to *Legs* overwhelms this combination.

Skipping over time 9, where a similar effect occurs, at time 10 there are partial matches to T1-T6 – including two "retrieval" episodes – but no complete match to any episode. Here, the match of two large-domain features to T1, along with its stronger temporal prior, lead to preferring it versus the more recent episodes.

At time 11, a "retrieval" episode (T8) is selected, as it is the matching episode with the highest temporal prior. However, the features retrieved here are a composite across multiple stored episodes, something that can happen whenever the retrieved episode has unspecified features. The value of 2 for *Legs* is retrieved from T8, but an automatic fallback also occurs, with T8's unspecified features retrieved from the best episode(s) having values for them (T1). This was not a planned feature of episodic memory, but a discovery about how this approach works.

An experiment was also performed with this domain on episodic replay given a retrieved starting point. Such replay is controlled rather than automatic in Sigma, involving the repeated selection and application of *replay operators* that increment `Time*Episodic` to prompt successive episodes. This works because the deliberate incrementing of `Time*Episodic` overwhelms the normal input from episodic selection. One oddity though is that, because incrementing here occurs via the general mental imagery operation of *translation* (Rosenbloom, 2011) – Sigma's native form of addition – it could as easily replay a sequence backwards, or even replay every third episode.

Sigma's episodic learning also works directly for less artificial pre-existing programs. For example, enabling it in the Eight Puzzle automatically tracks the state of the board, the goal (although it does not actually change), and the selected operator. This last bit enables automatically suggesting operators that might be useful in future situations. These results also illustrate how episodic learning works as well for complex relations as for simple feature-value pairs, and for continuous features as well as discrete (or symbolic) ones. The `board` predicate, for example, has three arguments, with each episode involving a distribution over the *tile* given continuous 2D locations.

The biggest issue with this overall approach to episodic memory is scalability. Focusing on changes helps, but even so, each update yields new messages that embody the entire memory. The resulting recomputations get expensive as the memory grows (Figure 12). A more scalable approach would leverage *incremental message passing*, where only regions that have changed – i.e., just those for the most recent time – are processed each cycle. This is an important future direction for Sigma's message passing algorithm.

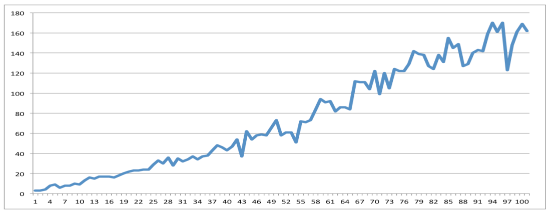

Figure 12: Time (msec) per decision across 100 decisions over a set of randomly generated predicates.

A smaller issue is the need to tweak several learning settings for episodic memory: the learning rate is set higher than normal for episodic features (to facilitate one-shot learning) and lower than normal for time (to handle its huge domain); and normalization during gradient descent here is divisive rather than Sigma's more typical subtractive (to learn an exponential temporal prior).

Conclusion

Episodic memory and learning has been implemented in a functionally elegant manner within Sigma; in particular, it has been deconstructed largely in terms of preexisting mechanisms. Added was a template-based generator for the predicates that form the episodic buffer in working memory and the conditionals that structure episodic long-term memory. Two settings in learning were adjusted as well. But with these modifications, automated episodic learning, selection and retrieval occurs each cognitive cycle.

The result is a compact episodic memory that records changes to state features, and a partial-match-based retrieval that prefers episodes as a function of both their matches to cues and a complex temporal prior. Retrieval can support simple replay, or specific aspects – such as operators – can be used more selectively in aiding decision-making.

Discoveries during this investigation include that: (1) the learned temporal prior naturally mimics base-level activation; (2) retrieval of partially specified episodes yields an automatic fall back to the best prior episode(s) with values for the missing features; and (3) the relative sizes of feature domains have an impact on the degree of match.

The two biggest items for future work are: incremental message passing for scaling of episodic memory; and exploring whether the combination here of template-driven structure generation plus gradient-descent learning can yield additional forms of learning that are essential to cognition.

Acknowledgments

This work was sponsored by the U.S. Army. Statements and opinions expressed may not reflect the position or policy of the United States Government, and no official endorsement should be inferred. Abram Demski suggested that fall back may be occurring during retrieval of partial episodes.

References

Altmann, E. M. & Gray, W. D. (1998). Pervasive episodic memory: Evidence from a control-of-attention paradigm. In *Proceedings of the 20th Annual Conference of the Cognitive Science Society* (pp. 42-47). Erlbaum.

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

Baddeley, A. (2000). The episodic buffer: a new component of working memory? *Trends in Cognitive Science*, *4*, 417-423.

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

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

Kolodner, J. (1993). *Case-Based Reasoning*. San Mateo, CA: Morgan Kaufmann.

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

Nuxoll, A. M. & Laird, J. E. (2012). Enhancing Intelligent Agents with Episodic Memory. *Cognitive Systems Research*, *17-18*, 34-48.

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

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). IOS Press.

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

Rosenbloom, P. S. (2013). The Sigma cognitive architecture and system. *AISB Quarterly*, *136*, 4-13.

Rosenbloom, P. S., Demski, A., Han, T. & Ustun, V. (2013). Learning via gradient descent in Sigma. *Proceedings of the 12th International Conference on Cognitive Modeling* (pp. 35-40).

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* (pp. 1146-1152). San Mateo, CA: Morgan Kaufmann.

Stracuzzi, D.J., Li, N., Cleveland, G. & Langley, P. (2009). Representing and reasoning over time in a unified cognitive architecture. *Proceedings of the 31st Annual Meeting of the Cognitive Science Society.*

Tulving, E. (1983). *Elements of episodic memory*. Oxford: Clarendon Press.

Vere, S., & Bickmore, T. (1990). A basic agent. *Computational Intelligence*, *6*, 41–60.