A Graphical Rethinking of the Cognitive Inner Loop

Paul S. Rosenbloom
Department of Computer Science & Institute for Creative Technologies
University of Southern California
rosenbloom@usc.edu

Abstract

Explorations of graphical representation and reasoning have yielded intriguing results spanning symbol, probability and signal processing. Here we explore an integrative application of graphs, as a path towards cognitive architectures of increased elegance, functionality, and extensibility. The specific focus is on steps towards a graphical reimplementation and extension of the cognitive inner loop within the Soar architecture. Alchemy, an implementation of Markov logic, is used for initial experiments, yielding insights into what will ultimately be required for full graphical implementations of enhanced cognitive inner loops.

1 Introduction

In [Rosenbloom, 2009] a new strategy was laid out for developing cognitive architectures [Newell, 1990] via a uniform *implementation level* based on factor graphs [Kschischang *et al*., 2001]. A cognitive architecture seeks to provide a coherent integration of capabilities sufficient for human-level artificial intelligence, whether in the context of a detailed model of human cognition or a system more loosely tied to the specifics of human behavior. Such an architecture requires the integration of a wide range of cognitive capabilities for, among other things, representation and memory, problem solving and planning, learning, reflection, interaction (including perception and motor control, use of language, etc), and the social aspects of cognition (such as emotion, collaboration, etc.).

The implementation level for cognitive architectures sits below the architectural memories and mechanisms, and provides the technologies out of which they are built. Traditionally, it is simply a programming language of some sort that may impact the efficiency, portability and robustness of the architecture, but is itself of little theoretical interest. The idea of basing the implementation level on graphical models looks to go beyond this by leveraging the uniform manner in which they support broad varieties of symbol, probability, and signal processing. The intuition is that, if the range of capabilities required for human-level intelligence can be built out of, and integrated within, such a uniform substrate then new architectures that are more elegant, functional and extensible may be possible. The ultimate goal is thus to determine whether a graphical implementation level can enable a new and improved generation of architectures.

As a step in this direction, I have been investigating a graphical reimplementation and enhancement of the Soar architecture [Rosenbloom *et al*., 1993]. Soar is one of the longest standing – over 25 years – and most thoroughly investigated cognitive architectures. It also possesses the unusual status of existing in both relatively uniform (up through version 8 [Laird and Rosenbloom, 1996]) and diverse (version 9 [Laird, 2008]) forms, providing a natural path for reimplementation that starts with a uniform version and then attempts a more uniform reintegration of later diversity. Simultaneously, opportunities can also be sought for expanding beyond Soar's predominant symbol processing paradigm, through the deep integration of probability and signal processing, in support of improved reasoning about, and interaction with, the real world.

This article: (1) examines what is involved in reconstructing a more uniform and functional graph-based *cognitive inner loop* for Soar, i.e., its core *decision cycle*, in which memory is accessed about the current situation and a decision is made about what to do next; (2) reports results from experiments towards this end based on Alchemy [Domingos *et al*., 2006]; and (3) identifies the path forward from here. While not yet achieving a fully implemented and enhanced version of Soar's decision cycle, it does yield critical insights into what will be necessary. First, however, the next two sections cover prior background on cognitive scales, Soar, and the graphical reimplementation of Soar.

2 Cognitive Scales and Soar

Part of the theory behind Soar as a model of human cognition is that *scale counts in cognition* [Newell, 1990]. As cognition is analyzed in depth, the phenomena and their properties change as the focus shifts from small spatiotemporal scales to larger ones. Newell discusses time scales from 10^{-4} seconds (100 μs) up to 10^{7} seconds (months), and divides them into four bands in human cognition: biological (10^{-4}-10^{-2} seconds), cognitive (10^{-1}-10^{1} seconds), rational (10^{2}-10^{4} seconds) and social (10^{5}-10^{7} seconds). In the biological band in particular there is also a spatial aspect to these scales, since signals are limited in how far they can travel within such small time frames. Organelles (10^{-4} seconds), neurons (10^{-3} seconds) and neural circuits (10^{-2} seconds) yield spatial scales within the biological band, before primitive deliberate acts (10^{-1} seconds) and operations (10^{0} seconds) are reached at the base of the cognitive band.

The architectural mechanisms in the earlier uniform versions of Soar were traditionally mapped onto a subset of these time scales, starting with the *elaboration cycle* at 10 ms (neural circuits), the decision cycle at 100 ms (deliberate acts), and activity in problem spaces at 1 second (operations) above this. The elaboration cycle involves parallel match (via a variant of the Rete algorithm [Forgy, 1982]) and firing of productions based on the contents of a global working memory. Functionally, it achieves one round of parallel associative retrieval of information relevant to the current situation. Production actions specify knowledge for potential retrieval while production conditions specify the circumstances under which that knowledge is relevant. Conditions also bind variables for use in actions.

The decision cycle involves repeated cycles of elaboration until quiescence; i.e., until no more productions can fire. This *elaboration phase* is followed by a decision based on preferences retrieved during elaboration. The elaboration phase yields an interpretation of the current situation, while the decision either selects an *operator* or generates an *impasse* if no operator can be selected. Impasses engender *reflection*, enabling processing to recur at the meta-level on the problem of making the decision. The decision cycle is Soar's cognitive inner loop – it accesses whatever knowledge is immediately available about the current situation and then attempts to decide what to do next.

A sequence of decisions yields activity in a problem space, amounting to some form of search if knowledge is limited and impasses occur. Search in problem spaces (ps-search) is: *slow*, with each decision occurring at the 100 ms level; *serial*, via a sequence of operator selections and applications; and potentially *combinatoric*, yielding trees that grow exponentially in the depth of the search. However, ps-search is open to control by any knowledge accessible during the decisions that occur within it. When the knowledge is sufficient to uniquely determine the outcome of each decision, behavior is more accurately characterized as algorithmic, or knowledge-driven, than as search.

Accessing knowledge during a decision can also be viewed as a search process – termed knowledge search (k-search) – but one that contrasts strongly with ps-search in character. K-search is: *fast*, with a 10 ms cycle time, *parallel*, both in match and firing of productions; and *subexponential*, at least in theory, if not in reality in most implementations. K-search occurs over a closed, extensionally defined, set of structures – the knowledge/productions in the system – rather than dynamically generating an open search space in the manner of ps-search. It is inherently algorithmic, rather than using an open cognitive loop, and is thus not itself penetrable by additional control knowledge.

Chunking [Laird *et al.*, 1986] is a learning mechanism in Soar that generates new productions based on the results of problem space activity during impasses. It compiles knowledge that is initially only available through activity at time scales of 1 second or more down to knowledge that is "immediately available" for use at the 10 ms time scale. Chunking, in combination with the flexibility of Soar's problem solving, has been shown to yield a much wider range of learning behaviors than just simple speed ups [Rosenbloom, 2006] – such as concept acquisition and episodic learning – but speeding up behavior remains its most essential functionality. In fact, the difficulty of producing some of these wider learning behaviors, and of integrating them with routine cognitive activity, was a key driver in Soar 9's shift towards diversity. Soar 9 adds, among other things, new varieties of long-term memory and learning.

3 Prior Work on the Elaboration Cycle

The work reported in [Rosenbloom, 2009] focused on reimplementing Soar's elaboration cycle (10 ms); and, in particular, on factor-graph algorithms for production match. Factor graphs in general provide a means of efficiently working with nearly decomposable functions of many variables. They arose in coding theory, where they underlie the surprisingly effective performance of turbo codes. They are similar to Markov networks (aka Markov random fields) in being undirected graphs with nodes that correspond to variables. However, in addition to variable nodes there are also factor nodes that represent functions over subsets of the variables. Factor nodes are analogous to clique potentials/weights in Markov networks, but they are directly incorporated as network nodes in factor graphs. Inference in factor graphs may be done through variations on the standard summary-product algorithm – a message passing approach that generalizes the more familiar (loopy) belief propagation algorithm in Bayesian networks [Pearl, 1983] – or via Monte Carlo methods.

Although Soar's Rete match algorithm could potentially be implemented directly via factor graphs, the focus of the prior work was on match algorithms arising more naturally from factor graphs. The investigation began with a straightforward, although ultimately naïve, approach. Working memory was represented as a three dimensional array of potential working memory elements, with one dimension for each of the three slots of a working memory element – object, attribute and value – and a value of one in every array cell for which the corresponding element was in working memory and zero otherwise. In the factor graph, variable nodes corresponded to production variables while factor nodes corresponded to conditions and actions. Match occurred via the summary-product algorithm, passing messages about the legitimate bindings of condition variables, and eventually converging on bindings for action variables.

Without going into the gory details, this initial approach raised generality, correctness and efficiency issues that ultimately led, through a sequence of optimizations and conceptual adjustments, to a new graphical match algorithm combining: (1) a junction-tree-like approach for graph construction, to enable the tracking of compatible combinations of bindings for different variables; and (2) an N-dimensional generalization of quad/octrees (called *exptrees* for lack of an existing term) for working memory and messages that enables uniform regions – i.e., regions in which all of the potential working memory or message elements are either present (one) or absent (zero) – to be matched without examining each element individually. The resulting match algorithm yielded correct results with dramatically reduced match times from the naïve approach. It also avoided creating the full production instantiations required by Rete, reducing the worst-case bound on match cost to exponential in the *treewidth* of a production rather than in the number of conditions in the production (as in Rete).

Beyond match, the remainder of the elaboration cycle consists of the firing of productions, empowering instantiated actions to add and delete working memory elements by flipping the corresponding array values from zero to one or vice versa. Once working memory is updated, the next elaboration cycle can begin.

4 Rethinking the Decision Cycle

In the work reported here, the focus has moved up to the decision cycle (100 ms) – Soar's cognitive inner loop – comprising an elaboration phase and a decision. This is the lowest level at which knowledge may affect decisions, at which multiple fragments of knowledge may be combined, and at which k-search may involve more than one cycle of match and firing. It is also the key scale at which extending Soar beyond strictly symbolic processing could lead to radically expanded functionality and at which it makes sense to begin considering incorporation of Soar 9's diversity.

Any reimplementation of Soar's elaboration phase must support its three core functions: (1) elaborating the description of the current situation in working memory based on relevant long-term knowledge; (2) generating operator preferences based on this elaborated working memory; and (3) altering working memory to reflect the application of selected operators. The first two functions are mostly monotonic, while the third is inherently non-monotonic. Overall, operation is similar to that of a truth maintenance system [Doyle, 1979], with operators determining the current assumptions and elaborations automatically asserting and retracting as these assumptions change.

Two additional constraints on the long-term knowledge must also be met by any reimplementation of the elaboration phase. The first constraint is that it must be capable of being processed in bounded time and space. Soar's production-based elaboration phase runs in time that is bounded by the *volume* of the elaboration phase – cost per production × number of productions × number of elaboration cycles. In reality, the second dimension is close to constant, as a suitably optimized Rete algorithm enables match time to remain close to constant with growth in the number of productions [Doorenbos, 1993]. However, the other two dimensions can be problematic. As mentioned earlier, the cost per production may be exponential in the size of the production. Even worse, the length of the elaboration phase can be infinite – new working memory elements can be generated on each elaboration cycle that lead to more productions firing in the next cycle. A reimplementation should at least avoid exacerbating these boundedness issues, and ideally improve on them (such as the prior work's improved match bound).

The second constraint is that the long-term knowledge must be learnable. Soar acquires productions via chunking, and Soar 9 adds other mechanisms to acquire its additional varieties of long-term knowledge; but satisfying this constraint in a graphical reimplementation is left to future work.

Beyond these two constraints, the uniform versions of Soar also lived with the constraint that all long-term knowledge must be cast as productions. Productions have the advantage that they are uniform, active, relatively flexible, and learnable. They also have a long successful history in cognitive modeling. Still, they have proven balky in dealing with both declarative and perceptual knowledge, ultimately leading to the elimination of this long held constraint in Soar 9 and the addition of three new long-term memories – two for declarative knowledge (semantic and episodic) and one for perceptual knowledge (visual imagery) – each with its own distinct variety of knowledge structures.

The approach explored here is not to eliminate the third constraint, but to replace it with one based on the varieties of knowledge structures efficiently implementable via graphical models. The hope is thereby to support a much wider range of functionality – including symbol, probability, and signal processing, as well as Soar 9's new kinds of knowledge structures – in a general yet uniform fashion.

The prior work discussed in Section 3 implemented a complete elaboration cycle. A straightforward elaboration phase is thus obtainable merely by repeating these cycles until quiescence is reached. While such an elaboration phase has been implemented, and initial ideas exist for extending it to continuous values and declarative memory, it has a serious flaw in only being able to propagate information forward across rule firings. Bidirectional information flow is needed for probabilistic information to propagate correctly across rules. It is also necessary for the implementation of *trellis* diagrams – in which a graph is composed of a linked sequence of identical subgraphs – such as the hidden Markov models used in speech recognition and other varieties of sequential signal processing.

The prior implementation supported bidirectional information flow within rules, and reused the same rule graph on each elaboration cycle – as is needed for a trellis – but the only linkage across cycles was implicit in the working memory elements generated during early elaboration cycles and matched on later ones. In addition to a graph for the generalized rules, a graph representing rule instantiations and the linkages among them may be needed to support bidirectional information flow across the rule instantiations generated within an elaboration phase.

In contrast to the elaboration phase, there are many fewer constraints on the decision procedure that follows it. Decisions in Soar were based on vote counting in a very early version, on symbolic preferences – acceptable, reject, better worse, etc. – in most versions, and on a combination of symbolic and (additive) numeric preferences in Soar 9. The key constraint on a reimplementation of the decision procedure is that all of the preferences accessed during the elaboration phase must be combined in an appropriate and tractable manner to yield either the selection of a unique operator or the detection of an impasse.

5 Progress towards a New Decision Cycle

The lack of bidirectional message passing across elaboration cycles in the existing implementation, in conjunction with a desire to better understand the utility of existing graphical languages – in particular those that already combine some forms of symbolic and probabilistic reasoning, such as Alchemy, BLOG [Milch *et al.*, 2007], and FACTORIE [McCallum *et al.*, 2008] – for implementing cognitive architectures, led to the decision to begin investigating the revision of Soar's decision cycle via such a language. Alchemy, which is based on combining first-order logic and Markov networks to form *Markov logic*, was ultimately selected because it: supports forms of both symbolic and probabilistic processing along with nascent signal processing [Wang and Domingos, 2008], provides an obvious approach to working with both rules and their instantiations, is publically available, runs on multiple types of computers, and has manuals, tutorials, and rapid response to emailed questions.[1]

To date, several small-scale experiments have been run with Alchemy: (1) re-implementing simple production systems that had previously been implemented via factor graphs; (2) adding a form of semantic long-term memory to the production memory; (3) exploring an implementation of the eight puzzle, one of the earliest tasks investigated in Soar [Laird and Newell, 1983] and the basis for early learning experiments with it [Laird *et al.*, 1986]; and (4) experimenting with trellis diagrams.[2]

In Alchemy, a *Markov logic network* (MLN) is defined via first-order predicates and formulas, with weights assigned to the formulas. The MLN is then compiled into a *ground Markov network* with binary nodes for each ground predicate, links among nodes that appear in common formulas, and features for each possible ground formula. Inference is performed on this ground Markov network, unless additional optimizations such as laziness (where grounding only occurs for variables that take on non-default values [Poon *et al.*, 2008]) or lifting (where multiple ground atoms are combined into single network nodes when they can be guaranteed to pass the same messages during belief propagation [Singla and Domingos, 2008]) are included.

The initial mapping of Alchemy to Soar's decision cycle focused on the first two functions of the elaboration phase: elaborating the current situation in working memory based on the contents of (a production-based) long-term memory, and generating preferences. Productions were represented as conditional formulas in an MLN file and the state of working memory at the beginning of the decision cycle was represented as evidence in an Alchemy database file. A single elaboration phase was then mapped onto a single invocation of Alchemy's inference procedure with this network and database.

The details of this mapping and the ensuing experiments are relatively uninteresting, so they are omitted here to conserve space. What is worth noting though are the implications of these experiments for a graphical reimplementation of Soar in particular, and a graphical implementation level for cognitive architectures in general. The most critical result is that the core of the mapping works, enabling a uniform elaboration phase that combines Soar's standard rule-based capabilities with probabilistic reasoning, simple trellises and semantic memory. The approach solves the aforementioned bidirectional, across rule, information flow problem by compiling the rules into a ground Markov network, and then performing inference in this ground network. Because nodes in this network correspond to working memory elements, and each such node links to every other element with which it coexists in a ground formula, the ground Markov network provides a single linked network for the entire elaboration phase. If the rules define a trellis, by repetition across elaboration cycles, bidirectional inference also occurs appropriately for it.

Another major result concerns the nature of production match under this mapping. Alchemy does not use inference in graphs to perform the equivalent of match for conditional formulas. Instead, match corresponds to Alchemy's extra-network process of compiling (first-order) Markov logic networks down to ground Markov networks. In essence, the Markov logic network corresponds to the definition of the production system while the ground Markov network corresponds to working memory elements (the ground nodes) and production instantiations (the ground formulas). In contrast to the prior implementation, working memory elements correspond to distinct nodes in this network rather than simply serving as the basis for messages among nodes.

Given that the goal is ultimately to implement a broadly functional cognitive architecture uniformly in graphs, Alchemy's match-as-compilation approach is problematic. A key question for future work therefore becomes whether it is possible to unify match – i.e., the computation of ground instances from first-order formulas – with the other necessary forms of inference into a single graph that is processed in a uniform manner, or whether it will be necessary to develop a dual graph/network approach in which match occurs via a first-order graph that generates, and is linked to, a ground graph in which the remaining inference occurs. Either way, the decision cycle will need to be extended from its current two stages to three: (1) compilation/match to generate a ground/instantiated network; (2) inference in the ground/instantiated network; and (3) decision making.

A final significant outcome is more conceptual, and concerns the general mapping between graphical systems and the hierarchy of cognitive scales, particularly as mediated by the mapping of both onto Soar. If the elaboration phase – which performs k-search (100 ms) – consists of the compi-

[1] Alchemy has also been explored in the Icarus cognitive architecture [Langley and Choi, 2006], with a focus specifically on the implementation of an inference component [Stracuzzi, 2009].

[2] Several of these experiments have been replicated with BLOG, but the results do not fundamentally alter the conclusions reported here based on Alchemy.

lation of, and inference in, a multi-layer ground network, then two important consequences follow:

1. The goal for a probabilistic first-order reasoner should not be a single uniform system capable of directly solving any problem no matter how complex. Instead, it should be bounded to the needs of k-search; e.g., only being capable of finding local minima in the solution space. Problems too complex to be solved in this manner would require a sequence of deliberate acts – i.e., steps in a problem space (ps-search) at the 1-second time scale – to move among local minima in search of a global minimum. Systems like Alchemy can get stuck in local minima [Stracuzzi, 2009], but according to this argument that is all a flat inference system should ever strive for. Reaching global minima in general requires a sequence of deliberate acts.
2. Cycles of message passing map onto the neural circuit (10 ms) scale. Functionally this implies that the 10 ms scale supports (only) local propagation of information, the 100 ms scale supports global propagation but (only) local minima, and global minima generally require time scales of 1 sec and above unless the problem is particularly simple or the system gets lucky.

Beyond these major implications, several smaller yet still interesting results have also been extracted from the mapping and resulting experiments:

3. Production systems utilize specific forms of non-monotonic reasoning, including an implicit closed-world assumption about the contents of working memory, and the ability to arbitrarily add and delete working memory elements. Such capabilities map awkwardly onto first-order reasoners, such as Alchemy.
4. Many production systems, Soar included, provide the ability to generate new unique symbols via production actions. Although such actions are local to individual productions, the process of checking uniqueness is a global activity that is difficult to implement through local message passing in a graph/network.
5. Exptrees served a role in the prior work that is analogous to what laziness and lifting achieve in Alchemy. The latter mechanisms eliminate unnecessary computation, either by avoiding the processing of default values or by grouping together items that can be treated the same. With exptrees, defaults are identified naturally and items are grouped by region if their values are identical. Exptrees appear to be a coarser approach, but it may ultimately be possible to bring these approaches more into alignment.
6. Experiments with simple trellises (linked repetitions) and semantic memory (encoded as ground atoms) have shown the feasibility of incorporating both within the decision cycle, but they involve computing most probable explanations (MPEs) rather than the marginals used for production match in the prior work to generate all instantiations. One possibility for the future is to localize the use of marginals to the generation of ground networks from first-order networks, and use MPE for all computations in the ground network.

Reflections on the first two of these smaller outcomes, in conjunction with the prior conclusion that the 10 ms scale only performs local propagation, has led to the conclusion that neither non-monotonicity nor the generation of new unique symbols should occur in individual productions (i.e., within an elaboration cycle). Non-monotonic reasoning has an implicit global aspect to it, given that the current answer is always dependent on nothing else being true that would overturn it. Operator implementation – the third function of the elaboration phase – and negated conditions in productions are examples of non-monotonic processing that thus should be banned from rules and moved up to the level of decision cycles. Generation of new unique symbols also involves an obvious global aspect.

Beyond the issues of non-monotonicity and symbol generation, limiting global information propagation to decision cycles and above implies that semantic memory, when defined in terms of finding the best match in memory to a cue [Anderson, 1990], should also occur at the level of decision cycles, as it currently does in Soar 9. Even more critically, though, this raises hard questions about the use of a global working memory in production match. One possible resolution to this dilemma would be to allow operator application to have global effects on working memory, as it is already being shifted up to the decision level, but to require elaboration to proceed via local propagation of information. Whether this can work, and more generally how to develop an effective architecture when all of the non-local forms of processing currently embodied by rules are moved up to the decision level, is a key issue for future work.

The actual decision making process has been neglected so far in this discussion. Limited experiments have been performed by leveraging Alchemy's provision of weights on formulas to encode preferences, and MPE inference to select operators based on these preferences. This has proven adequate for simple examples, but more complex ones are presently foundering on the preliminary step of dynamically generating operator instantiations and the accompanying unique symbols that are needed to identify them. Developing a full decision mechanism is thus left for future work.

6 Summary and Future Directions

This article has begun the exploration of graphical models for Soar's cognitive inner loop, with an Alchemy-based implementation of an elaboration phase that combines Soar's symbolic, rule-based, long-term memory with probabilities, simple bidirectional trellises and long-term semantic memory. In the process, four directions for the future have been explicitly called out: (1) satisfying the learnability constraint on long-term knowledge; (2) unifying rule match with inference in graphs while determining the respective roles of marginal versus MPE inference; (3) understanding how to feasibly and functionally move all non-local processing from the elaboration cycle to the decision cycle; and (4) implementing a complete decision procedure.

In addition, the full incorporation of signal processing, for perception and motor control, and of semantic and episodic knowledge is critical, and remains to be done. Beyond the inner loop, there is more of Soar to be explored, along with other existing architectures and hybridizations among them. Totally new architectures that take full advantage of what graphical models provide also need investigation. The ultimate intent is to definitively answer the question first posed in [Rosenbloom, 2009] as to whether implementing cognitive architectures on top of a uniform graph-based implementation level can yield a new generation of architectures with improved uniformity, functionality, and extensibility.

Acknowledgments

This effort was made possible by sabbatical support from the USC Viterbi School of Engineering plus funding from the Institute for Creative Technologies (ICT). ICT's Cognitive Architecture Working Group has been invaluable for semi-public exploration of these ideas. I would also like to thank the Alchemy group at the University of Washington for their help in installing Alchemy and working through various issues that arose during experimentation with it.

References

[Anderson, 1990] John R. Anderson. *The Adaptive Character of Thought*. Erlbaum, Hillsdale, NJ, 1990.

[Domingos *et al.*, 2006] Pedro Domingos, Stanley Kok, Hoifung Poon, Matt Richardson, and Parag Singla. Unifying logical and statistical AI. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 2-7, July 2006. AAAI Press.

[Doorenbos, 1993] Robert B. Doorenbos. Matching 100,000 Learned Rules. In *Proceedings of the 11th National Conference on Artificial Intelligence. Page 290-296,* 1993.

[Doyle, 1979] John Doyle. A Truth Maintenance System. *Artificial Intelligence*, 12(3): 251-272, 1979.

[Forgy, 1982] Charles L. Forgy. "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem". *Artificial Intelligence*, 19(1): 17-37, 1982.

[Kschischang *et al.*, 2001] Frank R. Kschischang, Brendan J. Frey, Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2): 498-519, February 2001.

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

[Laird and Newell, 1983] John E. Laird and Allen Newell. "A Universal Weak Method: Summary of Results." In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 771-773, Karlsruhe, FRG, August 1983. William Kaufmann.

[Laird and Rosenbloom, 1996] John E. Laird and Paul S. Rosenbloom. The evolution of the Soar cognitive architecture. In D. M. Steier. and T. M. Mitchell (Eds.), *Mind Matters: A Tribute to Allen Newell*, pages 1-50. Erlbaum, Mahwah, NJ, 1996.

[Laird *et al.*, 1986] John E. Laird, Paul S. Rosenbloom, and Allen Newell. Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning,* 1(1): 11-46, March 1986.

[Langley and Choi, 2006] Pat Langley and Dongkyu Choi. A unified cognitive architecture for physical systems. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. Boston, MA, 2006. AAAI Press.

[McCallum *et al.*, 2008] Andrew McCallum, Khashayar Rohanemanesh, Michael Wick, Karl Schultz and Sameer Singh. FACTORIE: Efficient probabilistic programming via imperative declarations of structure, inference and learning. In *Proceedings of the NIPS workshop on Probabilistic Programming*, Vancouver, Canada, 2008.

[Milch *et al.*, 2007] Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: Probabilistic models with unknown objects. In L. Getoor and B. Taskar, (Eds.) *Introduction to Statistical Relational Learning*, pages 373-398. MIT Press, Cambridge, MA, 2007.

[Newell, 1990] Allen Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA, 1990.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, San Mateo, CA, 1988.

[Poon *et al.*, 2008] Hoifung Poon, Pedro Domingos, and Marc Sumner. A general method for reducing the complexity of relational inference and its application to MCMC. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1075-1080, July 2008. AAAI Press.

[Rosenbloom, 2006] Paul S. Rosenbloom. A cognitive odyssey: From the power law of practice to a general learning mechanism and beyond. *Tutorials in Quantitative Methods for Psychology*, 2(2): 43-51, 2006.

[Rosenbloom, 2009] Paul S. Rosenbloom. Towards a new cognitive hourglass: Uniform implementation of cognitive architecture via factor graphs. Submitted to the *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[Rosenbloom *et al.*, 1993] Paul S. Rosenbloom, John E. Laird, and Allen Newell. *The Soar Papers: Research on Integrated Intelligence*. MIT Press, Cambridge, MA, 1993.

[Singla and Domingos, 2008] Parag Singla and Pedro Domingos. Lifted first-order belief propagation. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1094-1099, July 2008. AAAI Press.

[Stracuzzi, 2009] David Stracuzzi. Personal Communication, 2009.

[Wang and Domingos, 2008] Jue Wang and Pedro Domingos. Hybrid Markov logic networks. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1106-1111, July 2008. AAAI Press.