A Graphical Rethinking of the Cognitive Inner Loop

Paul S. Rosenbloom
Department of Computer Science
University of Southern California
rosenbloom@usc.edu

Abstract

Many varieties of graphical representation and reasoning have been explored, with interesting results spanning symbolic, probabilistic and signal processing. Here we explore an integrative application of graphs, as a step towards an ultimate goal of using them as the sole basis for implementing entire cognitive architectures. The specific focus in this article is a graphical reimplementation and extension of the *cognitive inner loop* within the Soar architecture; i.e., its decision cycle, where it cycles repeatedly through accessing knowledge relevant to the current situation and deciding what to do next. Alchemy, an implementation of Markov logic, is used for initial experiments, yielding key insights into what will eventually be required for a full graphical reimplementation and extension of Soar's cognitive inner loop.

1 Introduction

In [Rosenbloom, 2009] a new strategy was laid out for developing cognitive architectures [Newell, 1990] via a uniform implementation layer 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. Either way, 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 graphical approach to cognitive architecture looks in particular to leverage the uniform manner in which graphical representations and inference algorithms support a broad spectrum of symbolic, probabilistic, and signal processing algorithms. If the wide range of capabilities required for human-level intelligence can be built out of, and integrated within, such a uniform substrate, the hope is for new architectures that are both more elegant and more functional than the present generation.

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 architectures in current use. It also has the unusual status of existing in both relatively uniform (up through version 8) and diverse (version 9 [Laird, 2008]) forms, providing a natural path for reimplementation, starting with a uniform version and then attempting a more uniform reintegration of the later diversity. Simultaneously, opportunities can also be sought for expanding beyond Soar's predominant symbolic processing paradigm, through the deep integration of probabilistic 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 – that is, 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 the Alchemy system [Domingos *et al*., 2006]; and (3) identifies the path forward from here. But first, we will proceed briefly through some relevant background material on spatiotemporal scales in cognition, Soar, and prior work towards a graph-based reimplementation of Soar.

2 Cognitive Scales and Soar

Part of the theoretical background for Soar as a model of human cognition is the notion that *scale counts in cognition* [Newell, 1990]. As cognition is analyzed in depth, the phenomena of interest and their properties change as the focus shifts from very small spatiotemporal scales up to larger ones. Newell discusses time scales from 10^{-4} seconds (100 μs) up to 10^{7} seconds (months), and divides these time scales 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 most prominently there is also a spatial aspect to these scales, since signals in the brain are limited in how far they can travel within such small time frames. Newell mentions organelles (10^{-4} seconds), neurons (10^{-3} seconds) and neural circuits (10^{-2} seconds) as spatial scales within the biological band, before reaching primitive deliberate acts (10^{-1} seconds) and operations (10^{0} seconds) at the base of the cognitive band.

In the uniform versions of Soar, its architectural mechanisms 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 – termed an *elaboration phase* – followed by a decision based on those preferences generated/retrieved during the elaboration phase. The elaboration phase effectively 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, during which whatever knowledge is immediately available about the current situation is accessed and then a decision about what to do next is attempted based on this knowledge.

A sequence of decisions yields activity in a problem space, often amounting to some form of search if knowledge is limited and impasses occur. Search in problem spaces (PS-Search) is slow (with each decision underlying it being roughly at the 100 ms level), serial (involving a sequence of operator selections and applications), and potentially combinatoric (yielding trees that grow exponentially in the depth of the search), but it 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 be viewed as a search process as well – called Knowledge Search (K-Search) – but one that contrasts strongly with PS-Search in character. K-Search is fast (10 ms cycle time), parallel (both in match and firing of productions), and subexponential in complexity (at least in theory, if not in reality in most implementations). K-Search searches 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. K-Search is inherently algorithmic rather than proceeding via an open cognitive loop, and as such it is not itself penetrable by 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 Soar's problem solving flexibility, has been shown to yield a much wider range of learning behaviors than just simple speed ups [Rosenbloom, 2006], but speeding up behavior remains its most essential functionality. The difficulties involved in producing some of these variant learning behaviors, and in integrating them in with routine cognitive activity, was one of the key drivers in the shift to architectural diversity in Soar 9; where, among other things, new long term memories and associated learning mechanisms, were added.

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 based algorithms for production match. Factor graphs provide a general approach to efficiently computing 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 correspond to functions over subsets of the variables. Factor notes have analogues in Markov networks – potentials – but in factor graphs they are fully incorporated as nodes in the graph structure. 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 sampling algorithms.

The Rete match algorithm could potentially be implemented via factor graphs, but the focus of the prior work was instead on 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 the domain of 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 graph, variable nodes corresponded to production variables while factor nodes corresponded to conditions and actions. Match occurred via the summary-product algorithm, passing messages concerning the legitimate bindings of production variables, and eventually converging on bindings for the action variables.

Without going into the gory details here, this initial approach had a range of generality, correctness and efficiency issues that ultimately led, through a sequence of optimizations and conceptual adjustments, to a new graph-based match algorithm that combined: (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 representing working memory and messages, enabling uniform regions – that is regions in which all of the potential working memory or message elements were 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 consisted of the firing of productions, empowering instantiated actions to add new elements to working memory – by switching the corresponding array elements from zero to one – and to delete others by a reverse value switch. Once working memory was updated, the system was ready for the next elaboration cycle.

4 Rethinking the Decision Cycle

In more recent work, the focus has moved up to the decision cycle (100 ms) – Soar's cognitive inner loop – comprising an elaboration phase (a sequence of elaboration cycles until quiescence) and a decision. This is the lowest level at which knowledge may affect decision making, at which multiple fragments of knowledge may be combined, and at which the search of this knowledge 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 preferences for operator selection 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 operator application determining the current assumptions and, elaborations and preferences automatically asserting and retracting as assumptions change.

Two other 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 – that is, by the cost per production × the number of productions × the number of elaboration cycles. In reality, the second dimension is close to constant, as suitable optimized Rete algorithms enable match time to remain close to constant with growth in the number of productions [Doorenbos, 1993]. However, the other two dimensions can be problematic. The length of the elaboration phase can be infinite, as new working memory elements can be generated on each elaboration cycle that may then cause additional productions to fire in the next cycle. Also, as mentioned earlier, the cost per production may be exponential in the size of the production. In a reimplementation, the minimal goal should be to avoid exacerbating these boundedness issues. However, we should not merely be satisfied with non-maleficence if any way can be found to improve existing bounds, such as the prior work's lowering of the bound on production cost.

The second constraint is that the long-term knowledge must be learnable. Soar's productions are learned via chunking, and additional learning mechanisms have been added in Soar 9 to cope with the addition of new types of long-term knowledge. While this constraint is important, and needs to be kept in mind, the task of actually satisfying it will be delegated to future work.

Beyond these two constraints, the earlier 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 with 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 structure.

The approach being explored here is not to eliminate the third constraint, but to replace it with one based on the varieties of knowledge structures that can effectively be supported by factor graphs and their ilk. The hope is thereupon to support a much wider range of functionality – including symbolic, probabilistic, and signal processing, as well as the new kinds of knowledge structures incorporated into Soar 9 – all in a relatively uniform fashion.

The prior work discussed in Section 3 implemented a complete elaboration cycle. As such, an elaboration phase should be obtainable simply by repeating these cycles until quiescence is reached; that is, until no more productions are eligible to fire. While an elaboration phase can be implemented in such a manner, the approach has so far not proven to be easily extendable to the more broadly functional elaboration phase that is desired. An initial path to some of the requisite extensions seems clear; for example, building on the exptree representation of working memory and messages to handle continuous values in support of probabilities and signals, and basing declarative memory on goal-directed (backwards) access of condition-less productions. However, flexible bidirectional message passing across elaboration cycles has proven to be considerably more problematic.

Flexible bidirectional message passing across elaboration cycles is critical for supporting *trellis* diagrams – in which a network is composed of a linked sequence of identical subnetworks – such as are used in speech recognition and other forms of sequential signal processing; and to enable probabilistic information to propagate correctly across productions. Ideally there should be a single multi-layer network for the entire elaboration phase, in which messages can propagate both forward and backward, as needed. The current implementation doesn't support this. It reuses the same network on each elaboration cycle, as in a trellis, but with the only linkage across cycles being implicit in the working memory elements that are matched and generated by the productions. It would seem that in addition to a network representing the productions, a network representing instantiations of the productions, along with the linkages among these instantiations, is also required.

In contrast to the situation with the elaboration phase, there are many fewer constraints on the decision procedure that processes the preferences retrieved during the elaboration phase. Decision making in Soar was based on vote counting in a very early version, on symbolic preferences – acceptable, reject, better worse, etc. – in most other versions, and on a combination of symbolic and (additive) numeric preferences in version 9. The key constraint on a re-implementation 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 if a decision cannot be made.

5 Progress towards a New Decision Cycle

The difficulty of linking across elaboration cycles in the existing implementation, in conjunction with a desire to better understand the potential 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 use of such a language. Alchemy, which is based on combining first-order logic and Markov networks in the form of *Markov logic*, was ultimately selected because it: supports forms of both symbolic and probabilistic processing along with nascent signal processing [Wang and Domingos, 2008], matched fairly well a priori the way I was thinking about the decision cycle, is publically available, runs on multiple types of computers, and is supported with manuals, tutorials, and rapid response to emailed questions.

I have so far run several small-scale experiments with Alchemy: (1) re-implementing simple production systems that had been implemented in the prior work via factor graphs; (2) adding a form of semantic long-term memory to the production memory; (3) exploring a Soar-like implementation of the eight puzzle, one of the earliest tasks implemented 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.

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* that has a binary node for each possible ground predicate, links among nodes that appear in common formulas, and features for each possible ground formula. Inference is then 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 identified in Section 4, elaborating the current situation in working memory based on the contents of (a production-based) long-term memory and generating preferences. Straightforwardly enough, this mapping was based on representing productions as conditional formulas in an MLN file and representing the state of working memory at the beginning of the decision cycle 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 most compelling result from this mapping is that it directly solves the aforementioned across-elaboration-cycle linkage problem during creation of the ground Markov network. Because nodes in this network correspond to (potential) working memory elements, and each such node is linked 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. Moreover, when the firing of a production during one elaboration cycle triggers the same production, possibly with altered variable bindings, again on then next cycle, we have the basis for trellis diagrams.

A second major outcome concerns the nature of production match under this mapping. Alchemy does not use inference in graphs/networks to perform the equivalent of match for conditional formulas. Instead, production 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 (potential) working memory elements (the ground nodes) plus (potential) production instantiations (the ground formulas). Working memory elements correspond to distinct nodes in this network rather than simply serving as the basis for messages among nodes.

If the goal is to implement all of a broadly functional cognitive architecture uniformly in graphs – as my long-term goal indeed is – then Alchemy's approach of match-as-compilation is problematic. A key question for future work therefore becomes whether it is possible to unify match – or, more generally, the computation of ground instances from first-order formulas – with the other necessary forms of processing 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, meta-level graph that generates, and is linked to, a ground graph in which the remaining processing occurs. Under the latter approach, and possibly the former, the decision cycle becomes extended from its current two stages to three stages: (1) a compilation/match process that yields a ground/instantiated network; (2) inference in the ground/instantiated network; and (3) decision making.

A third significant outcome is more conceptual in nature, and concerns the general mapping between graphical systems and the hierarchy of cognitive scales outlined in Section 2, particularly as mediated by the mapping of both onto Soar. If the elaboration phase – which performs K-Search (100 ms) – consists of the compilation of, and inference in, a multi-layer ground network, then two important consequences follow.

First, based on the discussion in Section 4, it suggests that such inferences should be limited computationally. The goal should not be to develop a full probabilistic first-order reasoner as a uniform/flat system that is to be used directly for solving any problem, no matter how complex. Instead, there should be a bounded inference capability that simply defines the scope of K-Search. This might, for example, be capable of only reaching local minima in the search. Problems that are too complex to be solved in this manner would require a sequence of steps through successive local minima until a global minimum is found. In other words, reliable solution of complex problems would require a sequence of deliberate acts; i.e., activity in a problem space (PS-Search) at the 1 second and above scale. I have heard concerns that Alchemy may tend to get stuck in local minima; but if this mapping is right, that is all a flat inference system should ever strive for. The task of reaching global minima should require a sequence of deliberate acts, in which a system like Alchemy – possibly constrained even further computationally – may provide the inner loop.

Second, it suggests that message passing cycles may be mapped 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 (only) local minima, and that global minima generally require time scales of 1 sec and above unless the problem is particularly simple or the system happens to be lucky.

Beyond these major outcomes, several smaller yet still interesting results have also been extracted from the mapping and resulting experiments. In an abbreviated, itemized form, these are:

1. 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.
2. Many productions 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.
3. 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.
4. The addition of both trellises and semantic memory within the decision cycle is feasible, although it is not yet clear whether a single inference approach – of computing marginals versus most probable explanations (MPEs) – will ultimately be appropriate for all of these different types of knowledge structures.

Reflections on the first two of these smaller outcomes, in conjunction with the just prior conclusion that the 10 ms scale only supports local propagation, has led to the conclusion that neither non-monotonicity nor the generation of new unique symbols should be allowed in individual productions (i.e., at the 10 ms elaboration cycle scale). The argument with respect to non-monotonicity is tricky, and I won't attempt to lay it out in detail here, but the basic notion is that, at least in the forms in which it shows up in production systems, there is – or should be – a global aspect to it. Notice that this also implies that the third function of the elaboration phase, of implementing operators, should not happen via individual production firings during elaboration cycles, as currently occurs in Soar, but instead in some other form at the level of decision cycles. In contrast to non-monotonicity, generation of new unique symbols is a clear case in which there is a global aspect. (Semantic memory, which is usually defined in cognitive science in terms of finding the best match in memory to a cue [Anderson, 1990], is also a global process, and thus appropriately occurs at the level of decision cycles, as it is currently implemented in Soar 9.) Another item for future work is understanding how to get by without non-local capabilities at the elaboration cycle level of a system such as Soar, while exploring how such capabilities might instead legitimately and functionally be provided at the next level up (at the level of decision cycles).

The actual decision making process has been neglected so far in this section. I have been exploring an approach based on exploiting the weights Alchemy provides on formulas to encode preference information, and MPE inference to select operators based on this information. This works fine on simple examples, but more complex examples are foundering on the preliminary step of dynamically generating arbitrary operators during elaboration. In Soar, this generation depends on the previously discussed non-local capability for generating new unique symbols. Developing a full decision mechanism is left for future work.

6 Future Directions

Four directions for future work with respect to the graphical reimplementation and enhancement of Soar's inner cognitive loop have been explicitly called out in this article: (1) taking into account the learnability constraint on long-term knowledge; (2) the unification of match with inference in graphs; (3) understanding how to feasibly and functionally move all non-local processing from the elaboration cycle to the decision cycle; and (4) implementing decision making. 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 cognitive inner loop, there is more in Soar to be covered, as well as other existing cognitive architectures and possible hybridizations among them, not to mention the development of totally new architectures that take full advantage of what graphical models provide. In the process, the intent is to definitively answer the question posed in [Rosenbloom, 2009] as to whether implementing cognitive architectures on top of a uniform graph-based implementation level can improve 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 both in installing Alchemy and in working through a number of the 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 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 *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.

[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.

[McCallum *et al.*, 2008] Andrew McCallum, Khashayar Rohanemanesh, Michael Wick, Karl Schultz, 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.

[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.