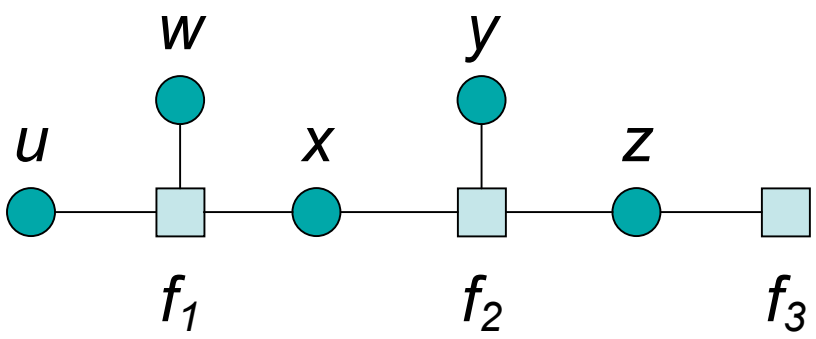
Brief Note on Factor Graphs for Cognitive Architecture

Paul S. Rosenbloom

October 24, 2008

<*** *Internal note, not for open distribution* ***>

Factor graphs (Kschischang, Brendan and Loeliger, 2001) provide a standardized way of representing and reasoning about complex functions of many variables in a semi-decomposable fashion. For example, a function of five variables, $f(u, w, x, y, z)$, may be decomposed into a product of factors each with fewer variables, such as $f_1(u, w, x)f_2(x, y, z)f_3(z)$, and then represented and reasoned about as a factor graph. A factor graph is a bipartite graph composed of variable nodes and factor nodes. There is one variable node for each variable in the original function and one factor node for each factor in the product representation of the function. Each variable node is connected via a bidirectional link to every factor in which it appears, and equivalently, each factor node is connected to each of the variables that appears in it.



Reasoning in factor graphs occurs via message passing, with each message conveying information about the possible values of the variable (node) participating in the message. For example, any message in either direction between variable node W and factor node f_1 will concern the possible values of variable W. In slightly more detail, a message from a variable to a factor conveys information the variable node has gathered from all of the other factors in which it participates about its possible values, while a message from a factor node to a variable conveys information the factor node has gathered from all of its other variables, in conjunction with the factor's own function, about the possible values of the variable to which the message is being sent. In the standard sum-product form of this algorithm, information about variable values is combined by multiplication at both variable and factor nodes, and then for factor nodes the implications of this combination for a single variable are determined by summing out over all other variables (thus the name "sum-product"). The generic version of the factor graph algorithm is not just limited to the combination of sum and product, but applies to any pair of operations that define a *commutative semi-ring*.[1] In addition to sum/product, other commutative semi-

[1] A commutative semi-ring is like a *ring*, except that the "multiplicative" operator must be commutative and elements need not have "additive" inverses.

rings can be based on max/product, OR/AND, and a variety of other operator combinations.

Factor graphs were developed originally in the context of coding theory, where they underlie the phenomenal performance of turbo codes. However, they have since been shown to be a generalization over many standard representations and state-of-the-art algorithms, such as loopy belief propagation in Bayesian networks, distributed arc consistency in constraint diagrams, the forward-backward algorithm in Hidden Markov Models (HMMs), the Viterbi algorithm, Kalman filtering, and the Fast Fourier Transform (FFT). I have also recently extended them to form the core of a simplified production system architecture, with a variant match semantics, in which it appears that the same representation and processing algorithm will yield both forward and backward chaining, along with interesting processing modes intermediate between these two (although only forward chaining has so far been implemented).

What is particularly notable about this range of applications of factor graphs is that they solve paradigmatic symbol processing (constraint diagrams and production systems), probability processing (Bayesian networks) and signal processing (HMMs, Viterbi algorithm, Kalman filtering, and FFTs) problems, and they do so via the same basic graphical representation and variations on the same message passing algorithm. The core difference in representation across these circumstances is simply: (1) whether the variable domains are discrete or continuous, and (2) whether the information about the variable values is Boolean (symbolic) or numeric (e.g., probabilities). Factor graphs also handle both static reasoning – drawing conclusions about a situation at a specific point in time – as well as dynamic reasoning across points in time. These various underlying commonalities have been noted in the literature by a number of researchers, and some have also tried to go beyond simple appreciation of this scope of uniformity to development of mixtures (of symbolic and probabilistic data and processing) and hybrids (of discrete and continuous variable domains), and even hybrid mixed models, both with and without dynamic/temporal characteristics (Gogate, 2007).

However, what no one has done to date is ask what the implications of this remarkable uniformity of representation and processing are for cognitive architectures. Cognitive architectures are systems that attempt to integrate together many, and ultimately all, of the domain-independent core representation/memory and processing capabilities required to create human-level intelligence (once the architecture has been supplemented with the requisite domain-specific knowledge). While such integration across capabilities is key for all cognitive architectures, some – such as Soar (prior to Soar 9) – also strive for a high degree of uniformity in their architectural mechanisms, while others – such as ACT-R, Soar 9 and Companion – provide a more heterogeneous collection of capabilities. *Factor graphs raise the possibility of developing a cognitive architecture – integrating symbolic and probabilistic/statistical inference and learning algorithms with effective (temporal) signal processing – that is highly functional yet remarkably uniform and simple.*

Heterogeneous capabilities may be constructed on top of this uniform factor-graph-based architecture – for example, for distinct procedural and declarative memories, and for visual and auditory processing systems – but they would all be grounded in this uniform factor-graph-based architecture. One interesting research strategy might be to determine to what extent it is possible to reimplement major existing architectures, such as the ones listed above, via a uniform factor graph architecture (which would be developed incrementally as the reimplementation needs of the other architectures are understood), and then to seek to go beyond them by: (1) looking at hybridization and simplification both within and across these architectures (for example, looking at whether a single factor graph can yield both the procedural and declarative memory capabilities in ACT-R and/or Soar 9, or whether Companion's analogy capabilities can be combined with the rule-based capabilities in Soar); and (2) integrating in new capabilities – such as perceptual and motor processing – that did not fit naturally into the existing architectures. It may then be possible to design a new architecture from the ground up, based on factor graphs at its core, that simply and elegantly provides a deep integration across more of the capabilities required for human-level intelligence than any existing architecture.

I am currently far from this ultimate goal, or even from the intermediate reimplementation and extension goals (other than having reimplemented a form of production system via factor graphs); however, this grand vision is what is driving my research program at this point.

Gogate, V. (2007). Approximate Inference in Probabilistic Graphical Models with Determinism, in *Doctoral Program of 22nd Conference on Artificial Intelligence (AAAI)*.

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