Factor Graphs

Continued…

Paul S. Rosenbloom

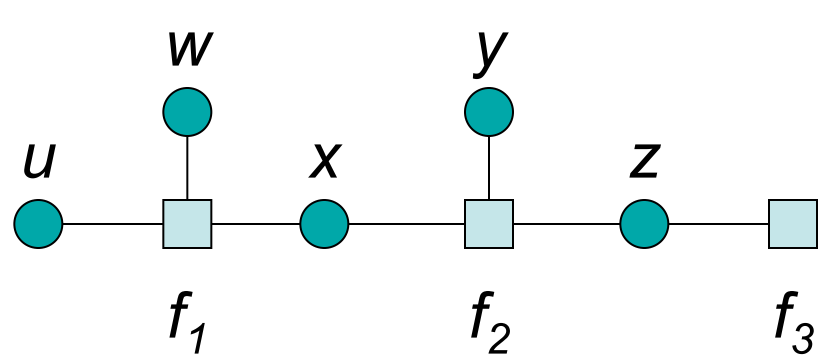
10/30/2008

Factor Graph Background

- Tame combinatorics in many calculations
  - Decoding codes (origin of factor graphs)
  - Bayesian networks and Markov random fields
  - HMMs (and signal processing more generally)
  - Constraint propagation
  - *Production match?*
- Form of divide-and-conquer with *nearly decomposable* components
- Many standard state-of-the-art algorithms can be derived from factor graph *sum-product algorithm*
  - Belief propagation in Bayesian networks
  - Forward-backward algorithm in HMMs
  - Kalman filter, Viterbi algorithm, FFT, turbo decoding
  - Equivalent to distributed arc-consistency in constraint diagrams
  - *Rete algorithm (or equivalent)?*

Structure of Factor Graphs

- Decompose functions into product of nearly independent *factors* (or *local functions*)
  - E.g., $f(u,w,x,y,z) = f_1(u,w,x)f_2(x,y,z)f_3(z)$
- Draw as bipartite graph
  - Nodes for factors and variables
  - Links between factors and their variables



Inference in Factor Graphs

- Reasoning occurs via message passing/propagation
  - From variable nodes to factor nodes
  - From factor nodes to variable nodes
- Each message from one node to a second node conveys some kind of information about the binding of a variable based on the information the first node has received from all of its neighbors other than the second node
  - Warning: A variable must take on a specific value
    - Simple setting of variable values
  - Belief: Weights/probabilities/potentials on domain of variable
    - This is the standard used in sum-product algorithm and Bayes nets
  - Survey: Likelihood of warning being required on domain elements
    - More effective than belief propagation
  - *Typical algorithms use only one of these uniformly*

Sum(mary)-Product Algorithm

- A variable node combines the messages from all of the factors to which it is connected (except for one to which message is to be sent)
  - Typically by point-wise multiplication of probabilities/potentials for each element of variable’s domain (*product*)

- A factor node combines information from all of the variable nodes to which it is connected (except for one to which message is to be sent) plus its own function
  - Also does a point-wise product, but in addition must marginalize out all of the variables not corresponding to the variable node to which the message is to be sent (*sum(mary)*)

Properties of Algorithm

- Applicable to any pair of operations defining a *commutative semi-ring*
  - Like a ring, but multiplication is commutative, and need not have an additive inverse
  - E.g., +/*, max/*, OR/AND
- Guaranteed to produce correct answer for *polytrees*
  - At most one undirected path between any two vertices
- Reduces to evaluation of expression trees for *trees* in which only care about value of root variable
- For graphs with loops, works well iteratively in many cases but not guaranteed to produce correct answer
  - May need to add a termination criterion as well
- Tied to concepts in statistical mechanics
  - Minimizes the Bethe free energy

Scope of FG and BP

Type of Variable Domains

Form of Messages	Discrete	Continuous
Boolean	Symbols	
Numeric	Probability (Distribution)	Signal & Probability (Density)

- Mixed models combine Boolean and numeric
  - For example, constraints and Bayesian networks
- Hybrid models combine discrete and continuous
- Hybrid mixed models combine all possibilities

Factor Graphs for Cognitive Architecture

- There is work on hybrid mixed models – although still very little and quite preliminary – but no one so far has looked at implications for cognitive architecture
- Idea/Hope/Fantasy: Factor graphs will provide a uniform layer for implementing and exploring cognitive architectures, while also pointing to novel architectures that are more uniform, integrated and functional
  - Yielding a better understanding of architectures and their modules/processes
  - Yielding hybrid mixed models that provide uniform integration
  - Potentially combining sequential and static reasoning
    - Dynamic hybrid mixed model (some work on this as well)
  - Generalizing STORM module interface approach
    - Rather than just setting interface variables (*warnings*), can send more subtle messages about their values (*beliefs*, *surveys*)

A Research Strategy

- Reimplement existing architectures via factor graphs
- Look to go beyond existing architectures by hybridization and simplification both within and across existing architectures
- Integrate in new capabilities that don’t fit well into existing architectures
  - E.g., vision and speech
- *I have started by looking at Soar*
  - *In particular, its production system architecture*

Factor Graphs for Production Systems

- Provides a space of alternative match algorithms
  - Vary in power and complexity
- Points in directions of (symbolic) extensions beyond simply forward chaining of rules
  - Backward chaining
  - Abduction
  - Constraint satisfaction
  - Analogy(?)
  - Combinations of approaches
- Provides insight in thinking about mixed models

Implemented Production System
(with some added syntactic sugar)

P1: Inherit Color

C1: (<v0> ^type <v1>)

C2: (<v1> ^color <v2>)

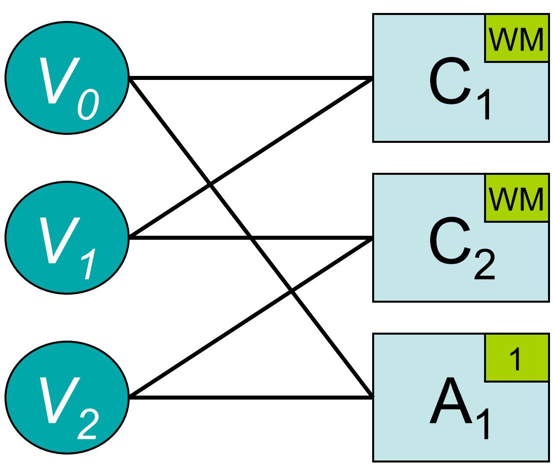
-->

A1: (<v0> ^color <v2>)

WM is a 3D Boolean array
- 1 when triple in WM
- 0 otherwise

Messages are Boolean vectors
- 1 when value possible
- 0 when value ruled out

$$P_1(v_0,v_1,v_2) = C_1(v_0,v_1)C_2(v_1,v_2)A_1(v_0,v_2)$$



WM (and goal constraints) are “sneaked” into factors

Only checks arc-consistency
Polynomial time

Variant Production System

WM and Goal via Nodes in Graph

P1: Inherit Color

C1: (<v0> ^type <v1>)

C2: (<v1> ^color <v2>)

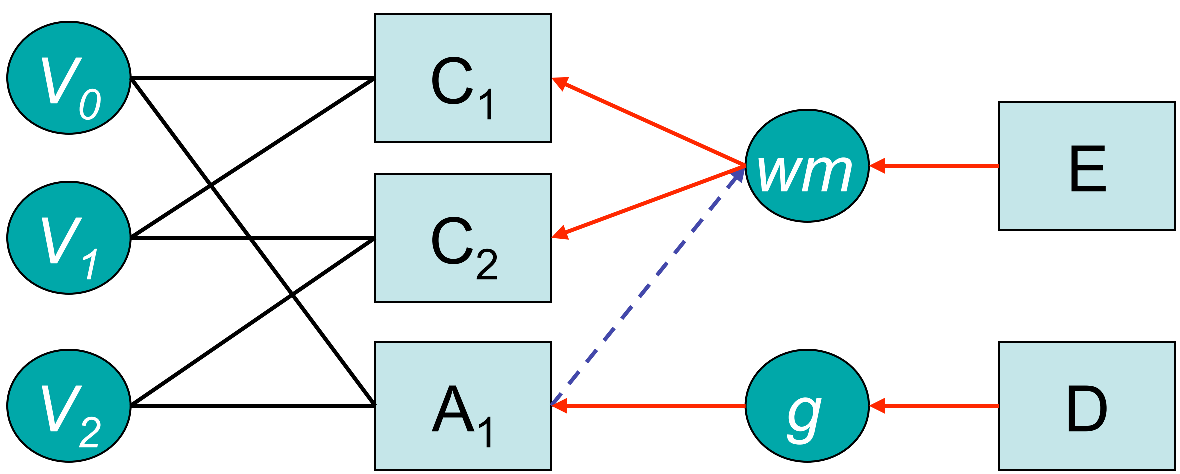
-->

A1: (<v0> ^color <v2>)

Solid arrows indicate no messages in other direction. Just computing deterministic functions of fixed evidence.

Dashed arrow says to change WM on next cycle*

$$P_1(\mathrm{wm}, v_0, v_1, v_2, g) = E(wm)C_1(v_0, v_1, wm)C_2(v_1, v_2, wm)A_1(g, v_0, v_1, wm)D(g)$$



Arc Consistency

P2: Path Confusion

C1: (<v0> ^type <v1>)

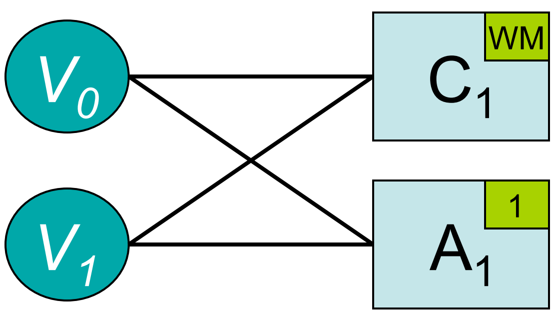
-->

A1: (<v0> ^type2 <v1>)

WM:

W1: (A ^type B)
W2: (C ^type D)

$$P_1(v_0,v_1) = C_1(v_0,v_1)\ A_1(v_0,v_1)$$



Match yields:

v_0 = {A, C}
v_1 = {B, D}

Action yields:

(A ^type2 B)
(A ^type2 D)
(C ^type2 B)
(C ^type2 D)

Called *instantiationless match* in earlier work

Some Possible Solutions

- Just live with it
- Divide action *weight* among ambiguous wmes
  - E.g., each new wme set to .25 rather than 1
  - Leverages potential mixed representation represent consequences of ambiguity
- Enforce path consistency by, e.g.
  - Extracting paths after generate binding sets
  - Implementing a factor graph that directly generates instantiations (ala Rete)
- Other approaches?

Rete in Factor Graphs

P1: Inherit Color

C1: (<v0> ^type <v1>)

C2: (<v1> ^color <v2>)

-->

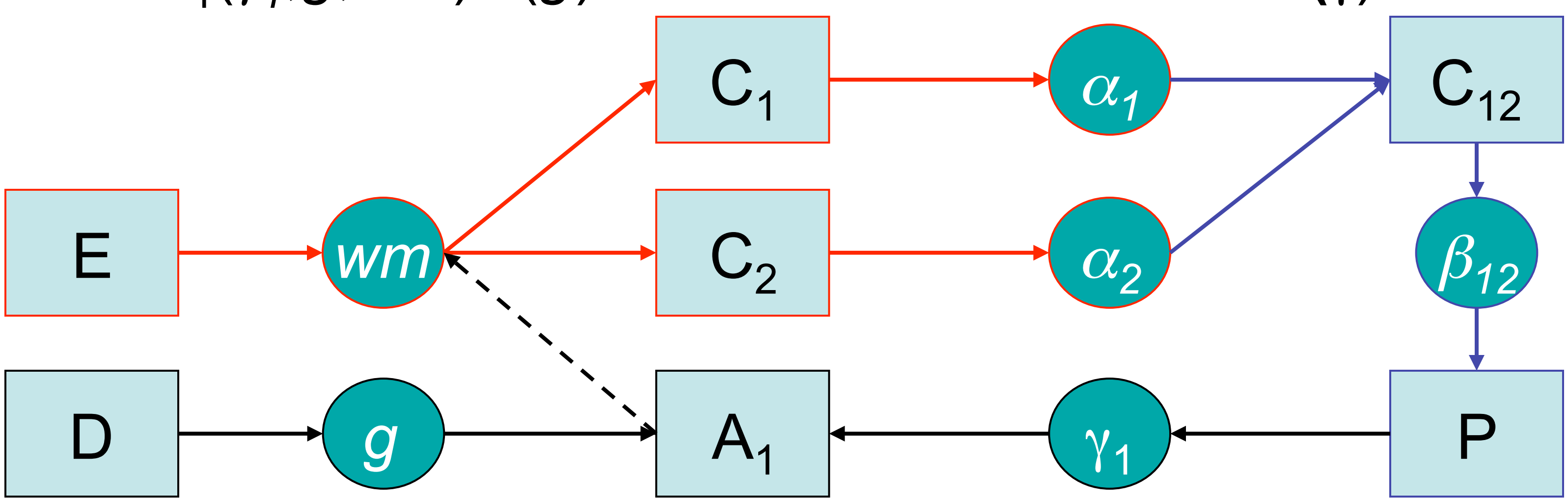
A1: (<v0> ^color <v2>)

$$\mathrm{P}_1(wm,v_0,v_1,v_2,g) = \mathrm{E}(wm)\mathrm{C}_1(wm,\alpha_1)\mathrm{C}_2(wm,\alpha_2)\,\mathrm{C}_{12}(\alpha_1,\alpha_2,\beta_{12})\mathrm{P}(\beta_{12},\gamma_1)\,\mathrm{A}_1(\gamma_1,g,wm)\mathrm{D}(g)$$

α network

β network

Action (γ) network



Comments on Rete in FG

- Combines Rete's α and β networks, plus actions (γ), into a single graph structure processed in a uniform manner
- Graph can be evaluated as an expression tree
  - Guaranteed solution with no iteration required
- Size of extensional β messages is 3^k for *k* conditions
  - Need to use sparse or hierarchical message structures
  - Sparse structure just sends elements that are 1 (instantiations)
  - Hierarchical structure does nested array region specification
    - E.g., start with stating that whole array is 0 and then specify which subregions are 1, but can also then nest this further with exceptions that are 0 for subsubregions, etc.
    - Generalization over standard sparse/instantiation-based representation in match that is more efficient when instantiations are clustered in array
    - Also transfers better to mixed case (used in some Bayes net approaches)

Beyond Forward Chaining

- Use rule definitions in new ways
  - Backward chaining
    - Use Goal array to constrain bindings of action variables to what want and thus indirectly constrain bindings of condition variables
    - Propagate constraint backwards by adding to goal wmes that will enable rules to fire in forward direction
  - Hybrid forward/backward chaining
    - Unconstrained goal array yields forward chaining
    - Tightly constrained goal array yields backward chaining
    - Moderately constrained goal array yields some kind of mixed behavior
  - Abduction
    - Allow backward chaining to change WM at select times
- Combine rule definitions with other symbolic graphs
  - Constraints are fully symmetric graphs
  - Facts and examples for declarative memory and analogy?
  - Others?

Bayesian Network (BN) Example

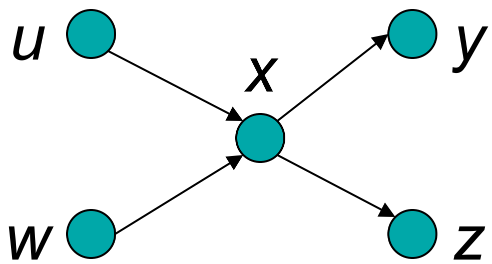
- Probabilistic reasoning involves computations of various quantities from *joint probability distributions* over random variables
  - E.g., compute the *marginal probability* $p(u)$ from the joint probability distribution $p(u,w,x,y,z)$ by summing out/over all of the other variables
    - $p(u) = \mathrm{SUM}_{w,x,y,z}\, p(u,w,x,y,z)$
  - Key for tractability is to do so without having to explicitly examine every combination of values of all of the other variables

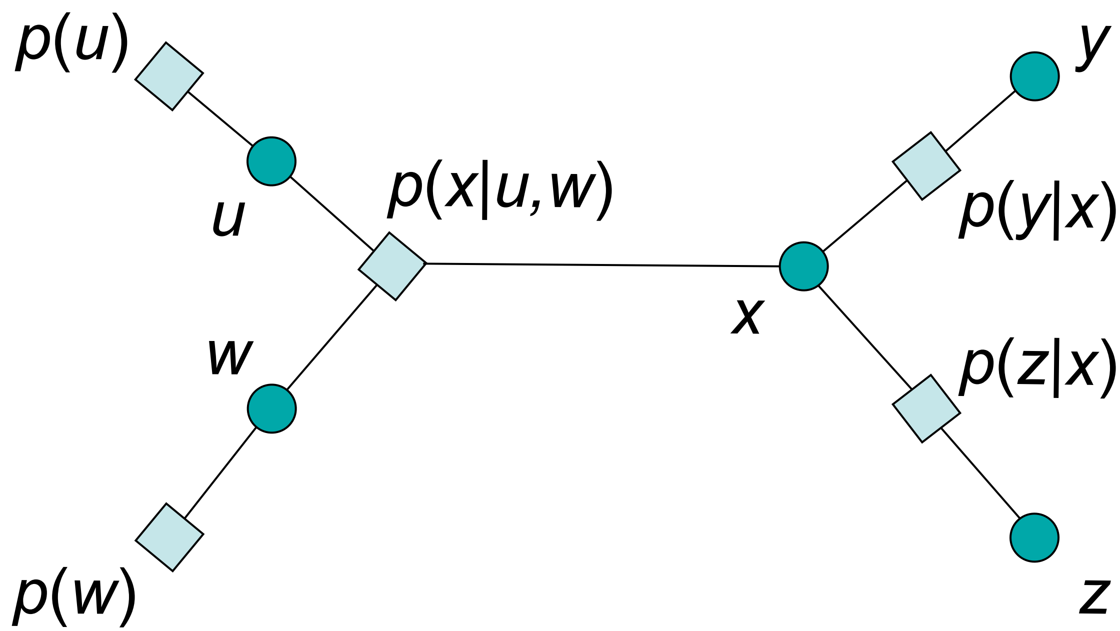
BN Example (cont.)

- A Bayesian network represents a joint probability distribution as the product of the conditional probabilities of each random variable
  - E.g., $p(u,w,x,y,z) = p(u)p(w)p(x|u,w)p(y|x)p(z|x)$
- Each conditional probability only involves a subset of the total set of variables (its *parents*)
  - Each variable is *conditionally independent* of all of the other variables, given its parents
    - I.e., once you know the values of the parent variables, the probability of a value of the variable can be determined independently of the values of all of the other variables
- Can radically reduce scope of combinatorics

Example BN and Factor Graph

$$p(u,w,x,y,z) = p(u)p(w)p(x|u,w)p(y|x)p(z|x)$$





Towards Mixed Graphs

- Existing work combines constraints and probabilities
  - Essentially hard and soft constraints
    - Hard constraints involve probabilities of 0 and 1 only
  - Standard BP in Bayesian networks works for such mixtures, but you can increase efficiency by preprocessing hard constraints
- What would mixtures do in our case?
  - Extend WM to be prior distribution on contents of WM
    - Move from Boolean to numeric values in array
  - Extend rules to be conditional probabilities (CPs)?
    - CP of wmes generatable by actions given wmes bound to conditions?
      - But may not provide full parents (in Bayes net sense) if other rules can generate same wmes
    - What is connectivity among these CPs in Bayes net sense?
      - Only part of domain of an action corresponds to part of a domain of another condition
      - Can we represent whole elaboration phase as a single Bayes net (trellis)?
  - Probabilities of rules being valid/accurate?
  - What else can be supported?
    - E.g., other probabilistic information, decision-theoretic decision making, statistical learning?