Graphical Models for Cognitive Architecture

Resolving the Diversity Dilemma

Paul S. Rosenbloom
Department of Computer Science & Institute for Creative Technologies
University of Southern California
June 26, 2009
The Diversity Dilemma

- Should an architecture’s mechanisms be uniform or diverse?
  - **Uniformity**: Minimal mechanisms combining in general ways
    - Appeals to simplicity and elegance
    - The “physicist’s approach”
    - The Challenge: Achieving full range of required functionality/coverage
  - **Diversity**: Large variety of specialized mechanisms
    - Appeals to functionality and optimization
    - The “biologist’s approach”
    - The Challenge: Achieving integrability, extensibility and maintainability
- Want best of both worlds, but a choice seems inevitable
  - Functionality tends to win, leading to the predominance of...
Example: Soar

- Through version 8 was a uniform architecture
- Version 9 has become highly diverse

Soar 3-8

Soar 9

Paul S. Rosenbloom 6/26/09
Proposal for Resolving the Dilemma

- Dig beneath architecture for uniformity at *implementation level* that supports diversity/functionality in architecture (and above)
  - Implementation level is normally just Lisp, C, Java, etc.
    - Impacts efficiency and robustness but usually not part of theory unless based on neural networks
- Base implementation level on *graphical models* for a uniform approach to symbol, probability and signal processing
  - Related to neural networks but broader
- Reconceive architectures via new implementation level
  - Reimplement, enhance and hybridize existing architectures
  - Develop new architectures
- Improve elegance, functionality, extensibility, integrability and
Graphical Models

- Efficient computation with multivariate functions
  - By decomposition over partial independencies
  - For constraints, probabilities, speech, etc.

- Come in a variety of related flavors
  - **Bayesian networks**: Directed, variable nodes
    - E.g., $p(u,w,x,y,z) = p(u)p(w)p(x|u,w)p(y|x)p(z|x)$
  - **Markov networks**: Undirected, variable nodes & clique potentials
    - Basis for *Markov logic* and *Alchemy*
  - **Factor graphs**: Undirected, variable & factor nodes
    - E.g., $f(u,w,x,y,z) = f_1(u,w,x)f_2(x,y,z)f_3(z)$

- Compute marginals via variants of
  - Sum-product (message passing)
  - Monte Carlo (sampling)
Potential for the Implementation Level

- State-of-the-art algorithms for *symbol*, *probability* and *signal* processing all derivable from the *sum-product algorithm*
  - Belief propagation in Bayesian networks
  - Forward-backward in hidden Markov models
  - Kalman filters, Viterbi algorithm, FFT, turbo decoding
  - Arc-consistency in constraint diagrams
- Potential to go beyond existing architectures to yield an effective and uniform basis for:
  - Fusing symbolic and probabilistic reasoning (mixed)
  - Unifying cognition with perception and motor control (hybrid)
  - Bridging from symbolic to neural processing
- Raises hope of a uniform implementation level that integrates broad functionality at the architecture level
## Scope of Sum-Product Algorithm

<table>
<thead>
<tr>
<th>Message/Variable Domain</th>
<th>Discrete</th>
<th>Continuous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Symbols</td>
<td></td>
</tr>
<tr>
<td>Numeric</td>
<td>Probability (Distribution)</td>
<td>Signal &amp; Probability (Density)</td>
</tr>
</tbody>
</table>

- **Mixed** models combine Boolean and numeric ranges
- **Hybrid** models combine discrete and continuous domains
- **Hybrid mixed** models combine all possibilities
- **Dynamic hybrid mixed** models add a temporal dimension
Research Strategy

Goals
- Evaluate extent to which graphical models can provide a uniform implementation layer for existing architectures
- Develop novel, more functional architectures
  - Enhancing and/or hybridizing existing architectures
  - Starting from scratch leveraging strengths of graphical models

Initial approach
- Reimplement and enhance the Soar architecture
  - One of the longest standing and most broadly applied architectures
  - Exists in both uniform (Soar ≤8) and diverse (Soar 9) forms
- Start from the bottom up, implementing uniform version while looking for opportunities to more uniformly incorporate Soar 9’s diversity plus critical capabilities beyond all versions of Soar
Progress to Date

- *Elaboration cycle* implementation via factor graphs
  - *Production match*
  - Production firing
- *Decision cycle* implementation via Alchemy (Markov logic)
  - *Elaboration phase*
  - Decision procedure
- With both also went beyond existing capability
  - *Lower complexity bound* for production match
    - Most recently, also began extension of WM beyond symbols
  - *Mixed* elaboration phase with simple *semantic memory* and *trellises*
- Still preliminary, partial implementations
  - Sufficient to demonstrate initial feasibility
  - Insufficient for full evaluation of impact on uniformity and functionality
Simple Mapping of Production Match onto Factor Graphs

P1: Inherit Color
  C1: \(<v_0> \text{^type} <v_1>\)
  C2: \(<v_1> \text{^color} <v_2>\)
  -->
  A1: \(<v_0> \text{^color} <v_2>\)

Model as a Boolean function:
\[ P_1(v_0, v_1, v_2) = C_1(v_0, v_1)C_2(v_1, v_2)A_1(v_0, v_2) \]

WM is 3D Boolean array \((obj \times att \times val)\)
1 when triple in WM
0 otherwise

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

Constant tests hidden in factors

WM is embedded in factors

Confuses binding combinations

May not check if rule completely matches

WM is 3D Boolean array \((obj \times att \times val)\)
1 when triple in WM
0 otherwise
Four issues have been resolved, yielding a new match algorithm
- Tracks variable binding combinations only as needed
- Complexity bound is exponential in \( treewidth \) rather than conditions
- Avoids some duplicate instantiations on a cycle
- Combines discrimination (\( \alpha \)) and join (\( \beta \)) activities in uniform graph

Solutions to binding confusion and rule matching increase number of rule variables processed at variable nodes
- Yields exponential growth in message size and processing cost
- Need to leverage tendency towards uniform values in WM and messages to reduce space and time costs
Hierarchical Memories and Messages

- N dimensional variant of quad/octrees (exptrees)
  - If entire space has one value, assign it to region
  - Otherwise, partition space into $2^N$ regions at next level, and recur
- WM & messages are piecewise constant functions
- Recently extended to piecewise linear functions
  - E.g., in 3D: $f(<x,y,z>, r) = A_r + B_{r,1}x + B_{r,2}y + B_{r,3}z$
  - Natural compact representation for probabilities, signals, images, etc.
    - Also handles symbols by setting the Bs to 0
- Implemented mem. but not yet all of sum-product
  - Product implemented with reapproximation
  - Support a spatial reasoning with time dimension?
  - Could also consider more adaptive...
Example Match Times

P1: Inherit Color
  C1: (<v0> ^type <v1>)
  C2: (<v1> ^color <v2>)
  -->
  A1: (<v0> ^color <v2>)

With solutions to all four problems, rule graph comprises 8 factor nodes and 8 variable nodes.

WM is $16^3$ in size, with 4 wmes

<table>
<thead>
<tr>
<th>Arrays</th>
<th>Sum of Products</th>
<th>Redistribute P over S</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Exceeded heap space</em></td>
<td>1.7 sec.</td>
<td>~7</td>
</tr>
<tr>
<td>132 sec.</td>
<td>.25 sec.</td>
<td>~500</td>
</tr>
</tbody>
</table>

Unoptimized Lisp
Implementing Soar’s Elaboration Phase via Alchemy (Markov logic)

- **Markov logic = First order logic + Markov networks**
  - Compiles weighted FOL into a ground Markov logic network
    - Node for each ground predicate
    - Weight for each ground clause (clique potentials)
      - Along with links among all nodes in ground clause
- **Goals for implementation**
  - Explore a *mixed* elaboration phase (rules & probabilities)
  - Explore semantic (fact) memory and *trellises*
  - Enable bidirectional message flow across rules
    - Normal elaboration cycle only propagates information forward
    - But need bidirectional settling for correct probabilities and *trellises*
  - Analogous to compilation of RL rules?
Encoding

- Convert productions into logical implications
  - Define types for objects and values of triples
    - colors={Red, Blue, Green} and objects = {A, B, C, D, E, F}
  - Define predicates for attributes
    - Color(objects, colors) and Type(objects, objects)
  - Specify implications/clauses for rules
    - (Type(v0, v1) ^ Color(v1, v2)) => Color(v0, v2).
  - Add weights to clauses as appropriate
- Initialize evidence (db file) with WM
  - Color(C, Red), Color(D, Blue), Type(A, C), Type(B, D)
- Semantic memory: weighted ground predicates: 10 Color(F, Green)
- Trellis: define via a pair of implications (accept & reject prefs.)
  - Size(step, size) => Size(step+1, size*2).
  - (Size(step, size1) ^ size1!=size2) => !Size(step, size2).
Alchemy Results

- Mapping basically works (modulo trellis strangeness)
  - Mixed representation with simple semantic memory and trellises
- Match occurs via graph compilation not message propagation
  - As Alchemy compiles first-order clauses to ground network
    - All symbolic reasoning in compilation and probabilistic in propagation?
  - Falls short of uniform processing in the graph itself
- Implies a three phase decision cycle
  1. Compile/match to generate a ground/instantiated network
  2. Perform probabilistic inference in the ground network
  3. Decide
- Exptrees yield variants of Alchemy’s *laziness* and *lifting*
  - Deal with *default values* and *groups of elements* processed in the same way

Paul S. Rosenbloom 6/26/09
Locality Implications

- Alchemy, and systems like it, get stuck in local minima
  - Generally considered a problem, but is it really?
- If Alchemy maps onto Soar’s decision cycle then it only needs to perform K-Search
  - Conceptualize K-Search functionally as yielding local minima?
  - If so, then finding global minima, in general, requires PS-Search
- Implication would be that Alchemy should just yield local minima, but it also needs PS-Search on top of it
  - The same might then be said for all one-level, logical and/or probabilistic inference systems
Locality Implications (cont.)

- Taking this a step further, we can hypothesize functionally that:
  - Elaboration Cycle (10 ms): Local propagation of information
  - Decision Cycle (100 ms): Global propagation but only local minima
  - Problem Space Search ($\geq 1$ sec): Global minima (via sequence of local minima)

- But this implies that the elaboration cycle can’t do global propagation of information
  - Explicit global: Creating unique identifiers
  - Implicit global: Non-monotonic (negated conditions, operator applications)
  - Accessing all of working memory?

- Could Soar function if global propagation were limited to the decision cycle?
  - I may need to answer this for a graphical implementation of Soar.
New approach to cognitive architecture
- Via a uniform graphical implementation level
  - Uncertain symbolic processing
  - Signal processing in inner loop
  - Potential bridge to neural
- May resolve diversity dilemma
  - Improving elegance, scope, integrability and maintainability
- Early results on elaboration cycle/phase are encouraging
  - New match algorithm with improved complexity bound
  - Mixed elaboration phase with semantic memory and trellises

Far from complete architecture
- Combine two experiments
- Add decisions, impasses, chunking
- Incorporate Soar 9 extensions
- Locality may be Achilles heel
  - Or mapping from mechanism to implementation may be so complex as to lose benefits of uniformity in implementation
- May be too slow for actual use
- A common implementation level need not guarantee clean integrability

Need to show not just more elegance, but increased utility...