Towards a Graphical Implementation Level for Cognitive Architecture

Resolving the Diversity Dilemma

Paul S. Rosenbloom

Department of Computer Science & Institute for Creative Technologies
University of Southern California

May 27, 2009
Outline

- Cognitive architecture
  - The diversity dilemma
- The implementation level
  - Based on graphical models
    Goal: A uniform implementation level for cognitive architecture supporting improved elegance, functionality, extensibility, integrability & maintainability
- Towards new architectures
  - A reimplementation of Soar and beyond
    Goal: A new architecture with improved elegance, functionality, extensibility, integrability & maintainability
Cognitive Architecture

- Fixed structure underlying cognition
  - Defines core memories, reasoning processes, learning mechanisms, external interfaces, etc.
- Yields intelligent behavior when combined with knowledge in memories
  - Including more advanced reasoning, learning, etc.
- May model human cognition, strive for human-like intelligence, or be purely artificial
- Closely related to agent architectures
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 diversity
  - But is there another way out?
Example: Soar

- Traditionally a uniform architecture
- Version 9 has become highly diverse
Proposal for Resolving the Dilemma

- Dig beneath the architecture for uniformity at the implementation level that supports diversity/functionality in the architecture (and above)
- Base implementation level on graphical models for a uniform approach to symbol, probability and signal processing
- Reconceive architectures via new implementation level
  - Reimplement, enhance and hybridize existing architectures
  - Develop new architectures with improved elegance, functionality, extensibility, integrability and maintainability
A cognitive architecture is a theory about the fixed structure at one or more systems levels

- Focused on **Cognitive Band**

The implementation level covers some region below architecture

- Focused on **Biological Band** in humans
- Can be elsewise in AI systems

- Across full hierarchy, diversity may remain constant or differ
  - Physicists and biologists expect same?
  - Networking assumes *hourglass*

(Newell, 1990)
The Internet hourglass

Applications

Web
FTP
Mail
News
Video
Audio
ping
napster

Transport protocols

TCP
SCTP
UDP
ICMP

IP

Link technologies

Ethernet
802.11
Power lines
ATM
Optical
Satellite
Bluetooth

From Hari Balakrishnan

5/27/09
Paul S. Rosenbloom
What About Cognition?

- Top (applications) is clearly diverse
  - Key part of what architectures try to explain
- Bottom is likely diverse as well
  - Physicalism: Grounded in diversity of biology
  - Strong AI: Also groundable in other technologies
- Is the waist **uniform** or **diverse**?
  - Hourglass or rectangle
  - Traditionally question about the architecture
What About Cognition?

- **Top (applications) is clearly diverse**
  - Key part of what architectures try to explain
- **Bottom is likely diverse as well**
  - Physicalism: Grounded in diversity of biology
  - Strong AI: Also groundable in other technologies
- **Is the waist uniform or diverse?**
  - Hourglass or rectangle
  - Traditionally question about the architecture
- **Now moving question down to implementation level**
  - Similar to Domingos’s call for an application layer in AI
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**: Und., variable nodes & clique potentials
    - Basis for *Markov logic* and *Alchemy*
  - **Factor graphs**: Und., variable & factor nodes
    - E.g., \( f(u,w,x,y,z) = f_1(u,w,x)f_2(x,y,z)f_3(z) \)
    - Processed via sum-product (message passing) or Monte Carlo (sampling) algorithms
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**
Factor Graph Sum-Product Algorithm

- Computes *marginals* (or MAP/MPE)

\[ f(x) = \Sigma_{u,w,y,z} f(u,w,x,y,z) = \Sigma_{u,w,y,z} f_1(u,w,x)f_2(x,y,z)f_3(z) = \Sigma_{u,w} f_1(u,w,x) \Sigma_{z} f_3(z) \Sigma_{y} f_2(x,y,z) \]

- Message passing about possible values of variables
  - Variable node combines messages from factors to which it is connected (except target factor)
    - Point-wise *product* of values for elements of variable’s domain
  - Factor node combines information from all variables to which it is connected (except target variable) plus its own function
    - Does a point-wise *product* as well
    - Also marginalizes out variables not part of target variable node (*sum*)
Properties of Sum-Product

- Applicable to any pair of ops defining a *commutative semi-ring*
  - Both ops are associative, commutative and have identity elements
  - The distributive law is defined: \( a(b+c) = ab+ac \)
  - *E.g.*, \(+/*\), \(\max/*\), \(\text{OR/AND}\)

- Behaves differently as function of structure of graph
  - Reduces to evaluation of expression trees for *tree-structured graphs* where only care about root
  - Guaranteed to produce correct answer for *polytrees*
    - At most one undirected path between any two vertices
  - 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 statistical mechanics, leading to extensions
  - Minimizes the Bethe free energy
### Scope of Sum-Product Algorithm

<table>
<thead>
<tr>
<th>Message/Variable Range</th>
<th>Message/Variable Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Discrete</td>
</tr>
<tr>
<td></td>
<td>Symbols</td>
</tr>
<tr>
<td>Numeric</td>
<td>Continuous</td>
</tr>
<tr>
<td></td>
<td>Probability (Distribution)</td>
</tr>
<tr>
<td></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
Diagrammatic View of Uniform Soar
## Hierarchical View of Uniform Soar

<table>
<thead>
<tr>
<th>Scale</th>
<th>Functionality</th>
<th>Mechanism</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 sec</td>
<td>Reflective</td>
<td>Problem Space Search</td>
<td>Impasse/Subgoal if can’t Decide</td>
</tr>
<tr>
<td>100 ms</td>
<td>Deliberative</td>
<td>Decision Cycle</td>
<td>Preference-based Decisions upon Quiescence</td>
</tr>
<tr>
<td>10 ms</td>
<td>Reactive</td>
<td>Elaboration Cycle</td>
<td>Parallel Rule Match &amp; Firing</td>
</tr>
</tbody>
</table>
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
- 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

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

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
P2: Binding Confusion

C1: \(<v0> ^\text{type} <v1>\)

-->

A1: \(<v0> ^\text{type2} <v1>\)

\[ P_2(v_0, v_1) = C_1(v_0, v_1) A_1(v_0, v_1) \]

\[ WM:\]

W1: (A ^\text{type} B)
W2: (C ^\text{type} D)

Match yields:

\[ v_0 = \{A, C\} \]
\[ v_1 = \{B, D\} \]

Action yields:

(A ^\text{type2} B)
(A ^\text{type2} D)
(C ^\text{type2} B)
(C ^\text{type2} D)

Called \textit{instantiationless match} in earlier work
Some Possible Solutions

- Alter processing to yield full $f(v_1, v_2, \ldots)$
  - Extract instantiations after generate independent binding sets
  - Alter factor graph to directly generate instantiations (à la Rete)

- Redefine what production match is to achieve
  - Define match to generate what factor graph yields: $f_1(v_1), f_2(v_2), \ldots$
    - Write rules so that this altered semantics is sufficient
  - Variation: Divide action weight among ambiguous wmes in WM
    - E.g., each new wme set to .25 rather than 1
    - Leverages potential mixed representation to handle ambiguity
  - Variation: Compute enough of full function for correct action bindings
    - Eliminate confusion by tracking variable combinations as needed
      - Related to variable stretching and junction trees
    - Eliminate redundant instantiations that have same action variable bindings and different condition variable bindings
Tracking Variable Combinations

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

\[ P_1(v_0, v_1, v_2) = C_1(v_0, v_1)C_2(v_1, v_2)A_1(v_0, v_2) \]
Tracking Variable Combinations

\[ P_1(v_0, v_1, v_2) = C_1(v_0, v_1)C_2(v_1, v_2)A_1(v_0, v_2) \]

Order factor nodes
Add intervening variable nodes
Compute where each rule variable is first and last used
Tracking Variable Combinations

\[ P_1(v_0, v_1, v_2) = C_1(v_0, v_1)C_2(v_1, v_2)A_1(v_0, v_2) \]

\[ P_1(v_0, v_1, v_2) = C_1(v_{01})C_2(v_{01}, v_{02})A_1(v_{02}) \]

Order factor nodes
Add intervening variable nodes

Compute where each rule variable is first and last used

Stretch rule variables over all nodes between first and last usage

Complexity bounded by max rule variables at a node (treewidth)
- Versus number of conditions in Rete

The approach works, but increases the combinatorics over the simple mapping
Message (& WM) Size Problem

- 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
- WM may itself be quite large: $max\text{-}symbols^3$
- Need to take advantage of the tendency towards uniform values in order to reduce size and processing cost
  - Both WM and messages are nearly all 0 or 1
- Solution: A hierarchical approach to nested region specification
Hierarchical Memories and Messages

- N dimensional generalization of quad/octrees (exptrees)
  - If entire space has one value, just assign it to the region
  - Otherwise partition space into $2^N$ regions at next level, and recur
- Views WM (and messages) as piece-wise constant functions
  - Could also conceivably be extended to piecewise linear functions
  - Natural compact representation for probabilities, signals, images, etc.
- Could alternatively do more adaptive partitioning that clusters/reorders points into regions based on patterns in the data

![Hierarchical Memory Diagram](image-url)
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></th>
<th>Product before Sum</th>
<th>Redistribute P over S</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Arrays</strong></td>
<td>Exceeded heap space</td>
<td>1.7 sec.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>~7</td>
</tr>
<tr>
<td><strong>Hierarchies</strong></td>
<td>132 sec.</td>
<td>.25 sec.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>~500</td>
</tr>
</tbody>
</table>
Implementing Soar’s Elaboration Phase via Alchemy (Markov logic)

- **Markov logic** = First order logic + Markov networks
  - 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)
  - Enable bidirectional message flow across rules
    - Normal elaboration cycle only propagates information forward
    - But need bidirectional settling across elaboration cycles and *trellises*
  - Explore semantic (fact) memory and *trellises*
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 database 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).
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

- Use of exptrees is analogous to Alchemy’s laziness and lifting
  - Deal with both default values and elements that can be processed in the same way
  - Exptrees may be cruder, but extension to piecewise linear functions is intriguing for dealing with continuous quantities
Speculations from Mapping to System Levels

If Alchemy maps onto Soar’s elaboration phase – and ultimately its full decision cycle – then:

Soar’s decision cycle needs to be expanded to three phases
1. Compile/match to generate a ground/instantiated network
2. Perform probabilistic inference in the ground network
3. Decide

Perhaps systems like Alchemy should not seek global minima
- Elaboration Cycle (10 ms): Local propagation of information
- Decision Cycle (100 ms): Global propagation but local minima
- Problem Space Search ($\geq$ 1 sec): Global minima (via sequence of locals)

Global processing should not happen within the elaboration cycle
- Creating unique identifiers in rules, non-monotonic processing, etc.
- Raises questions about extent to which WM should be global as well
Summary and Future

- A new kind of application for graphical models
  - Build *architectures* instead of more specific domain solutions or more general toolkits

- Reconceiving cognitive architecture via a graphical implementation level to
  - Resolve the diversity dilemma
  - Develop architectures with improved elegance, functionality, extensibility, integrability and maintainability
    - Combine symbolic and probabilistic reasoning
    - Unify cognition and perception
    - Bridge from symbolic to neural processing

- Exploring a graphical reimplementation and enhancement of Soar
  - Completed preliminary investigations of key parts of elaboration & decision cycles
    - Raises interesting issues about uniformity and locality of processing
  - Need to cover full decision cycle and then extend to reflection and learning
  - Looking to include probabilities and signals (for vision & speech) in the inner loop
  - Add Soar 9’s capabilities for semantic/episodic memory & reinforcement learning

- More broadly, need to
  - Reexamine other cognitive architectures and hybrids among them
  - Experiment with radically new architectures enabled by graphical models
  - Evaluate feasibility & utility of uniform implementation level for architectures
Automated Construction via Contour Crafting: A House in a Day (Behrokh Khoshnevis)