A Graphical Memory Architecture

Paul Rosenbloom
University of Southern California

5/20/2010

USC

The projects or efforts depicted were or are sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM) Simulation Training and Technology Center (STTC). The content or information presented does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

ICT
INSTITUTE FOR CREATIVE TECHNOLOGIES

Memory Architecture

- **Nature of memories used w/in decision cycle**
- **Short-term/working and long-term memories**
  - Soar 1-8: working memory + production memory
  - ACT-R: buffers + production memory, semantic memory
  - Soar 9: working memory, ST visual imagery + production memory, semantic memory, episodic memory, LT visual memory
- **Focus here is on representation and access**
  - Haven't yet got to learning

Goals

- **Broadly functional memory architecture**
  - Both procedural and declarative knowledge
  - *Hybrid*: Continuous/signal + discrete/symbolic
  - *Mixed*: Probabilistic/uncertain + discrete/symbolic
- **Uniform implementation**
- **Provide core for development of full hybrid mixed architecture**
  - Melding scope with simplicity and elegance

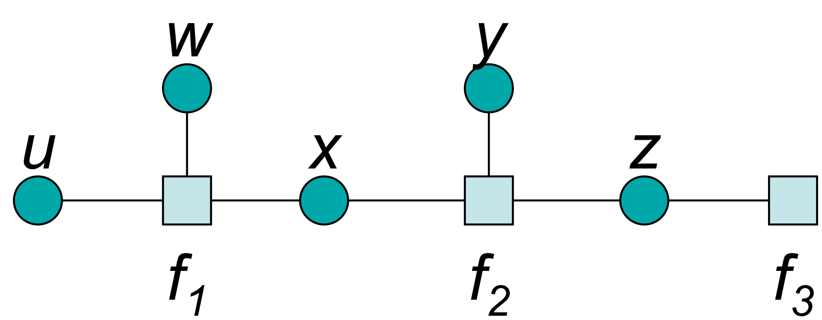
Approach

- **Base roughly on Soar 9 and ACT-R**
  - Working memory
  - Procedural LT Memory
    - Productions
  - Declarative LT Memory
    - Semantic: Predict unspecified attributes of objects based on specified ones (cues)
    - Episodic: Retrieve best episode based on recency and match to cues
  - Eventually imagery as well, but not yet
- **Implement via *graphical models***
  - Layered approach: *graph* and *memory* layers

Graph Layer: Factor Graphs w/ Summary Product

- **Factor graphs are undirected bipartite graphs**
  - Decompose functions: e.g., $f(u,w,x,y,z) = f_1(u,w,x) f_2(x,y,z) f_3(z)$
  - Map to variable & factor nodes (with functions in factor nodes)
- **Summary product algorithm does message passing**
  - Compute values of variables (marginals) by sum-product
  - Compute best overall (max. a posteriori) by max-product

Complete reimplementation from last year with improved functionality, generality, efficiency



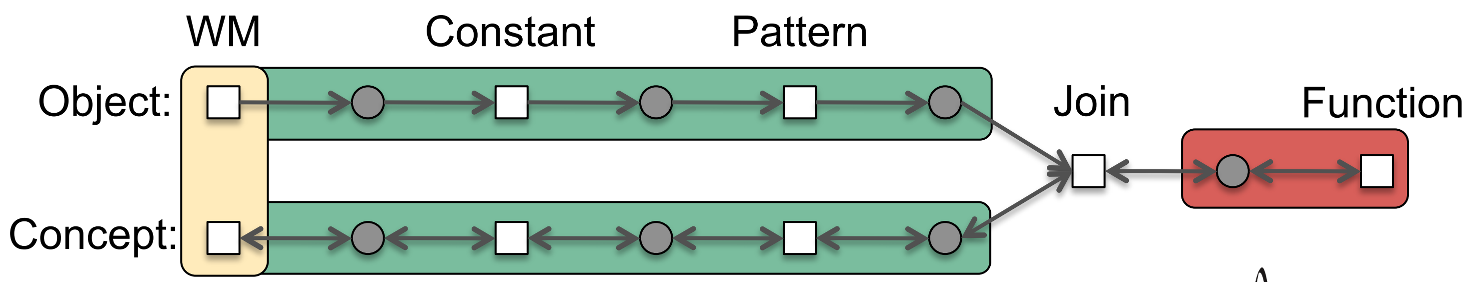
Generalized Function/Message Representation

- **N dimensional continuous functions**
  - Approximated as piecewise linear functions over rectilinear regions
- **Span (continuous) signals, (continuous and discrete) probability distributions, symbols**
  - *Discretize domain* for discrete distributions & symbolic
    - [0,1>, [1,2>, [2,3>, …
  - *Booleanize range* (and add symbol table) for symbolic
    - E.g., [0,1>=1 ➔ RED true; [1,2>=0 ➔ GREEN false

*y**x*	[0,10>	[10,25>	[25,50>
[0,5>	0	.2*y*	0
[5,15>	.5*x*	1	.1+.2*x*+.4*y*

Memory Layer: Distinguish WM and LTM

- **Representation is predicate based**
  - E.g., `Object(`*s*`,O1)`, `Concept(O1,`*c*`)`
  - Arguments may be constants, or variables (in LTM)
- **Long-term memories compile into graphs**
  - LTM is composed of *conditionals* (generalized rules)
  - Each conditional is a set of predicate patterns and a function
- **WM compiles into functions in peripheral factor nodes**
  - It is just an N dimensional continuous function where normal symbolic wmes correspond to unit regions with Boolean values



USC

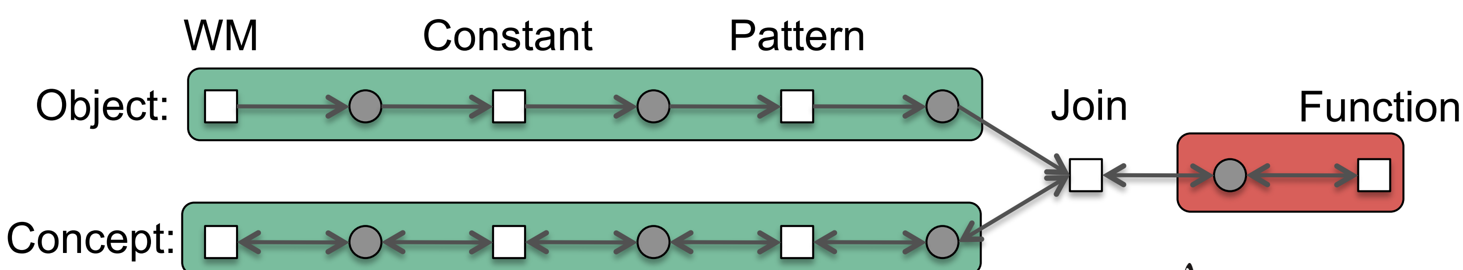
INSTITUTE FOR CREATIVE TECHNOLOGIES

Conditionals

```
CONDITIONAL ConceptPrior
   Condition: Object(s,O1)
   Condact: Concept(O1,c)
```

Walker	Table	Dog	Human
.1	.3	.5	.1

- **Patterns can be *conditions*, *actions* or *condacts***
  - Conditions and actions embody normal rule semantics
    - Conditions: Messages flow from WM
    - Actions: Messages flow towards WM
  - Condacts embody (bidirectional) constraint/probability semantics
    - Messages flow in both directions: local match + global influence
  - Encoded as (generalized) linear *alpha networks*
- **Pattern networks joined via bidirectional *beta network***
- **Functions are defined over condact variables**



USC
ICT
INSTITUTE FOR CREATIVE TECHNOLOGIES

Additional Details

- **Link directionality is set independently for each link**
  - Determines which messages are sent
- **Whether to use *sum* or *max* is specified on an individual variable/node basis**
  - Overall algorithm thus mixes sum-product and max-product
- **Variables can be specified as *unique* or *multiple***
  - Unique variables sum to 1 and use *sum* for marginals: [.1 .5 .4]
  - Multiple variables can have any or all elements valued at 1 and use *max* for marginals: [1 1 0 0 1]
- **Predicates can be declared as *open world* or *closed world* with respect to matching WM**
- ***Pattern variables* cause sharing of graph structure**
  - May be within a single conditional or across multiple conditionals

USC

INSTITUTE FOR CREATIVE TECHNOLOGIES

Memories

Production Memory

- Just conditions and actions
  - Although may also have a function
- CWA and multiple variables

```
CONDITIONAL Transitive
    Condition: Next(a,b)
               Next(b,c)
    Action: Next(a,c)
```

Semantic Memory

- Just condacts (in pure form)
- OWA and unique variables
- Naïve Bayes (prior on concept + conditionals on attributes)

```
CONDITIONAL ConceptWeight
    Condact: Concept(O1,c)[α1]
             Weight(O1,w)
```

w\c	Walker	Table	…
[1,10>	.01w	.001w	…
[10,20>	.2-.01w	"	…
[20,50>	0	.025-.00025w	…
[50,100>	"	"	…

```
CONDITIONAL ConceptPrior
    Condition: Object(s,O1)
    Condact: Concept(O1,c)[α1]
```

Walker	Table	Dog	Human
.1	.3	.5	.1

Memories (cont.)

Episodic Memory

- Just condacts (in pure form)
- OWA and unique variables
- Exponential prior on time + conditionals on episode attributes

```
Conditional TimeConcept
     Condact: Time(t)[α3]
              Concept(O1,c)
```

t\c	Walker	Table	Dog	Human
1	1	0	0	0
2	0	0	0	1
3	0	0	0	1
4	0	0	1	0

Constraint Memory

- Just condacts (in pure form)
- OWA and multiple variables

```
CONDITIONAL TwoColorConstraint12
   Condact: Color(R1,c1)[α7]
            Color(R2,c2)[α8]
```

c1\c2	Red	Blue
Red	0	1
Blue	1	0

```
CONDITIONAL TimePrior
     Condact: Time(t)[α3]
```

0	1	2	3	4
0	.032	.087	.237	.644

Key Similarities and Differences

Similarities

- All based on WM and LTM
- All LTM based on conditionals
- All conditionals map to graph
- Processing by summary product

Is analogy vs. generalization driven by max vs. sum over instance-based memory?

Differences

- Procedural vs. declarative
  - Conditions/actions vs. condacts
    - Directionality of message flow
  - Closed vs. open world
  - Multiple vs. unique variables
- Semantic vs. episodic
  - Marginal/sum vs. MAP/max
  - Condition on concept vs. time
  - General probs. vs. instances

Constraints are actually hybrid: condacts, OWA, multiple
Other variations and hybrids are also possible

Summary

Gold

- Uniform implementation of four distinct LTMs
- Reveals subtle underlying differences among the LTMs
- An important step towards a full hybrid mixed architecture
  - Working on decisions
  - Then subgoaling & learning
  - Proposal on imagery
    - Leverage continuous functions at core and known facility of graphical models for perception

Coal

- Subtle incompatibilities imply less uniformity in details
  - They have also proven quite difficult to resolve cleanly
- Progress can be slow & difficult
  - With occasional bursts of insight
- Not full memory implementations
  - And no learning
- Still far from full architecture
  - And from showing that there is a significant functional gain from this approach