# Semantic Memory in Soar

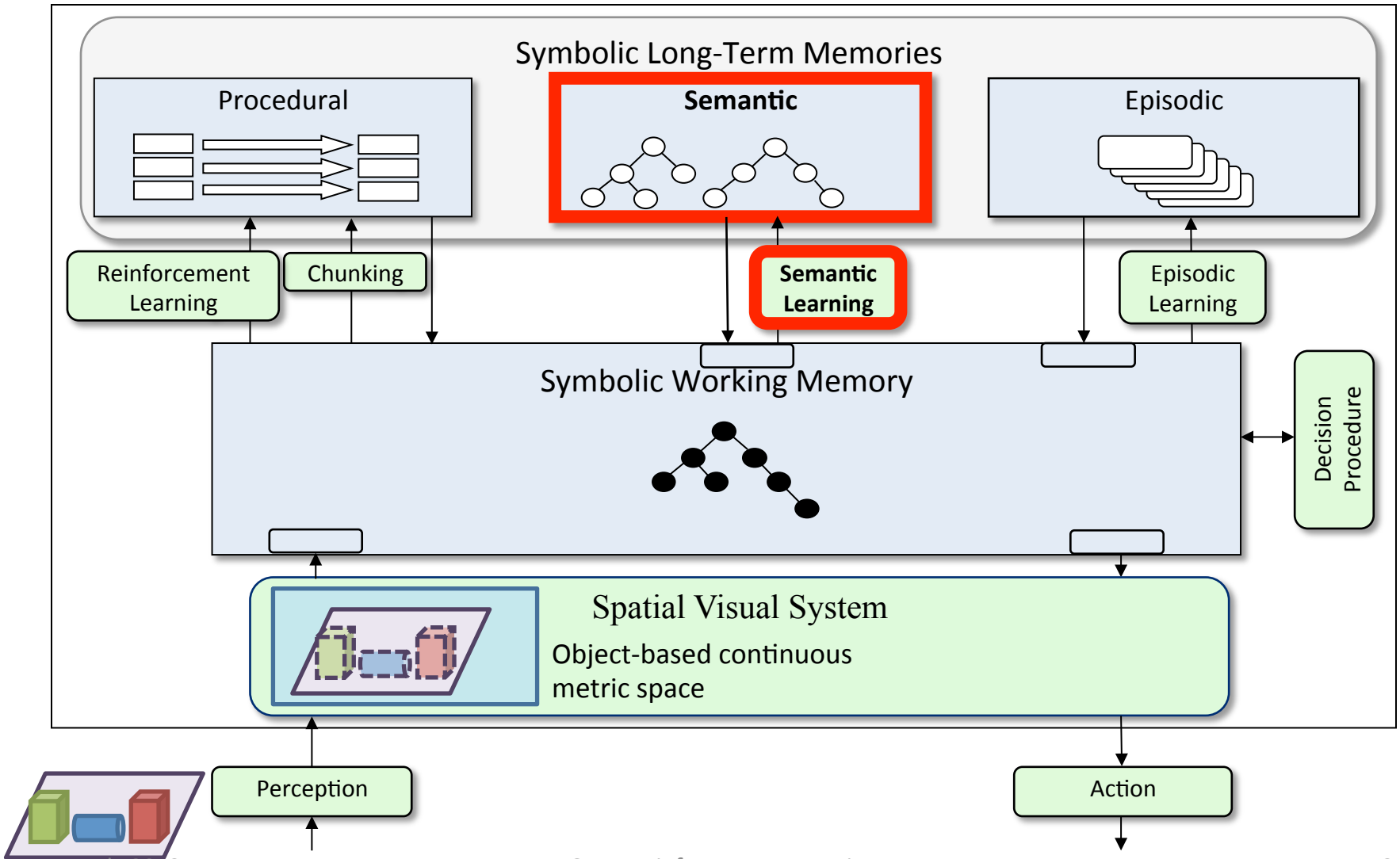
**Nate Derbinsky**

Disney Research, Boston

# Topics

- Semantic memory as a learning mechanism
- Basic usage
  - Example: multi-column arithmetic
- Scaling real-time performance
  - Task: word-sense disambiguation
- Comparison to ACT-R

# Soar 9

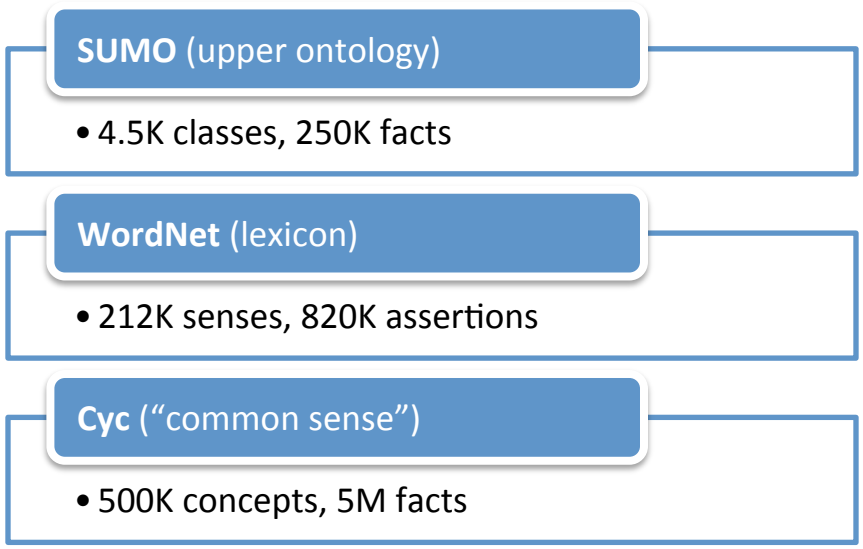

Symbolic Long-Term Memories
Procedural
Semantic
Episodic
Reinforcement Learning
Chunking
Semantic Learning
Episodic Learning
Symbolic Working Memory
Decision Procedure
Spatial Visual System
Object-based continuous metric space
Perception
Action

# Semantic Memory

Long-term store of general facts and relations about the world, independent of the context in which they were originally learned

## Agent Benefits

- Access to large KBs
- Retrieval bias as a reasoning heuristic (more on this later)



# Semantic Memory

## *Integration*

### Representation

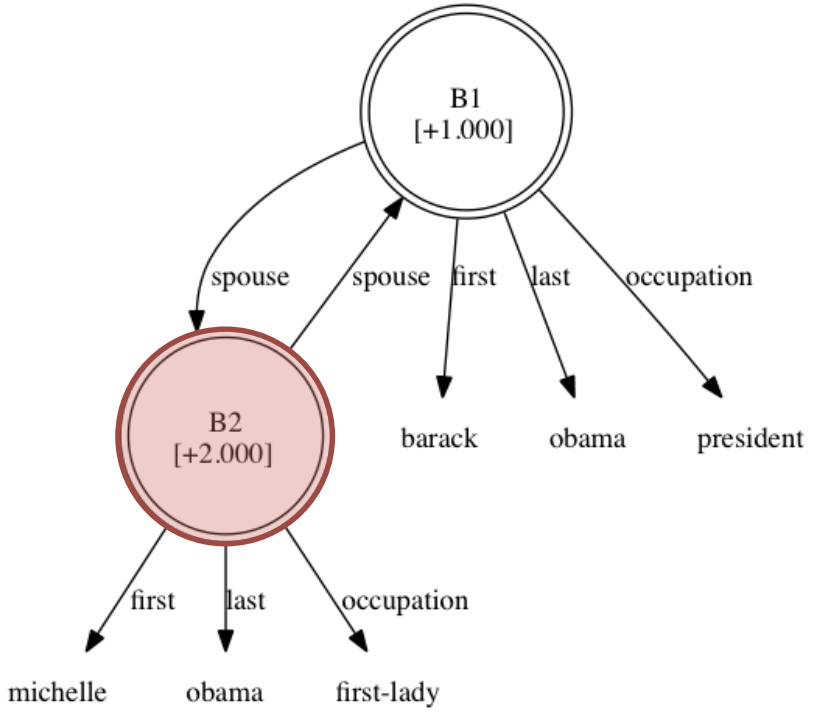
- Directed graph

### Encoding/Storage

- Incremental
- Deliberate

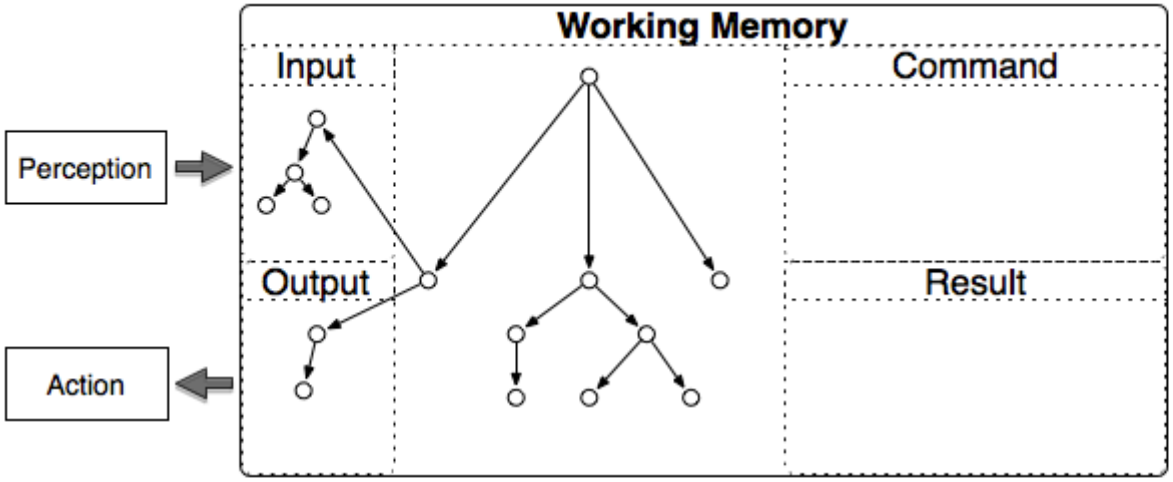
### Retrieval

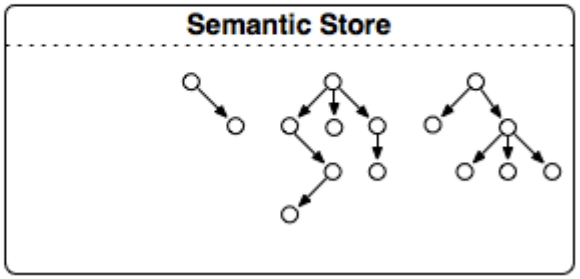
- Cue: set of features/relations
- Semantics: subset query
- Single result, ranked by bias value [#]




Example cue:
`last(obama),spouse(X)`

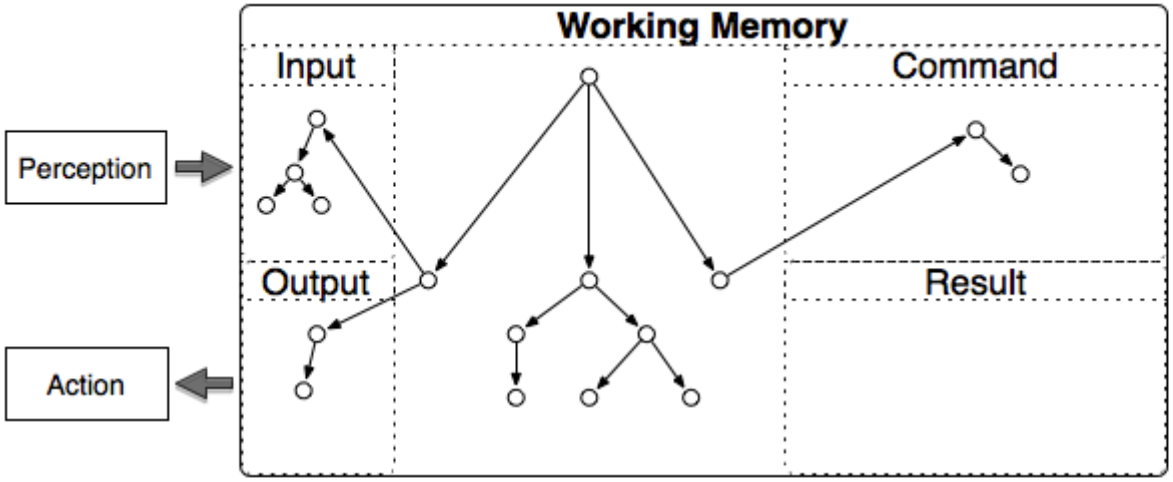
# Architectural Integration

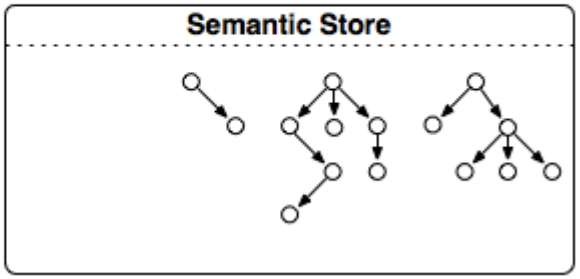


# Architectural Integration

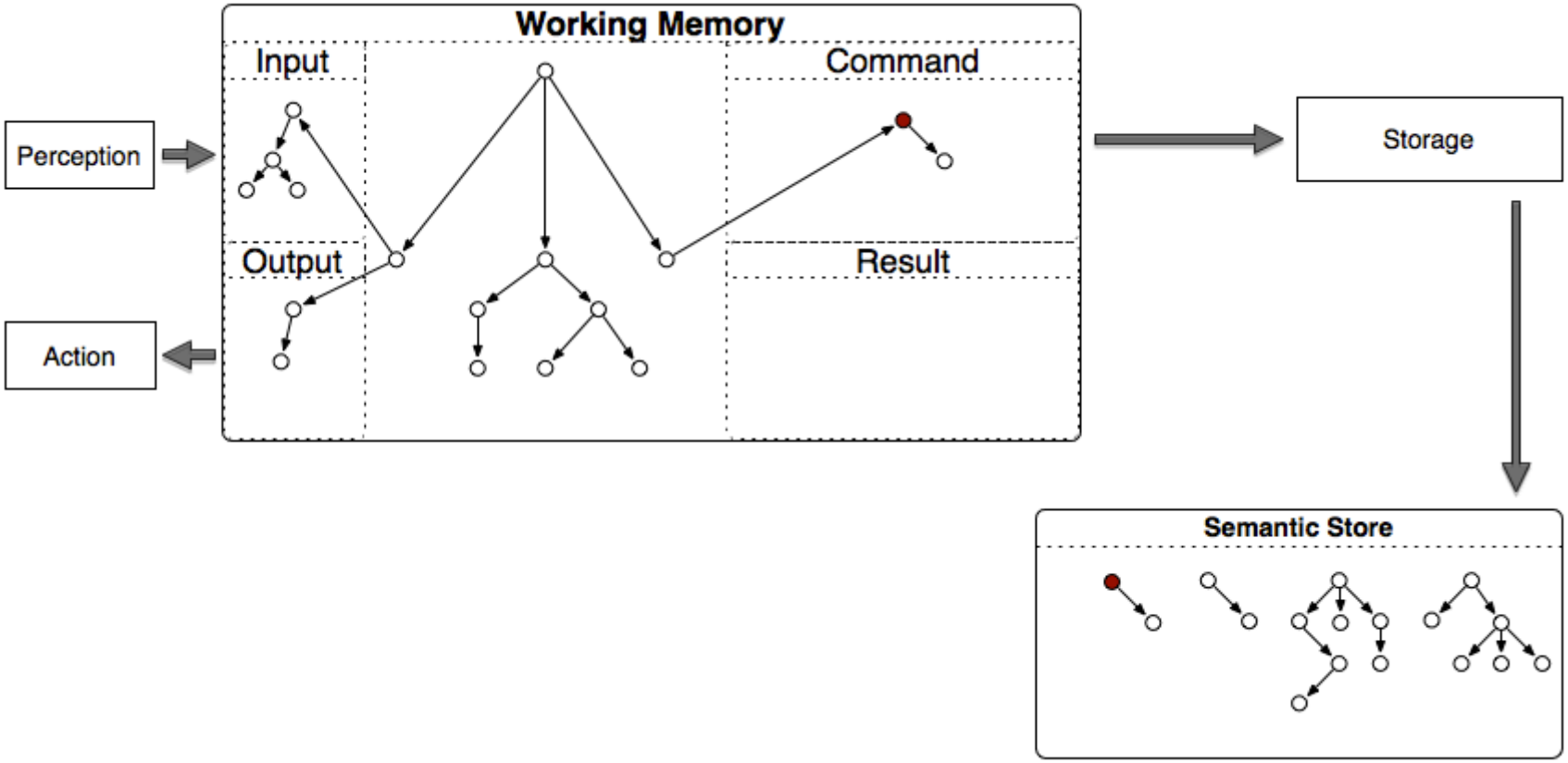
## *Deliberate Storage*



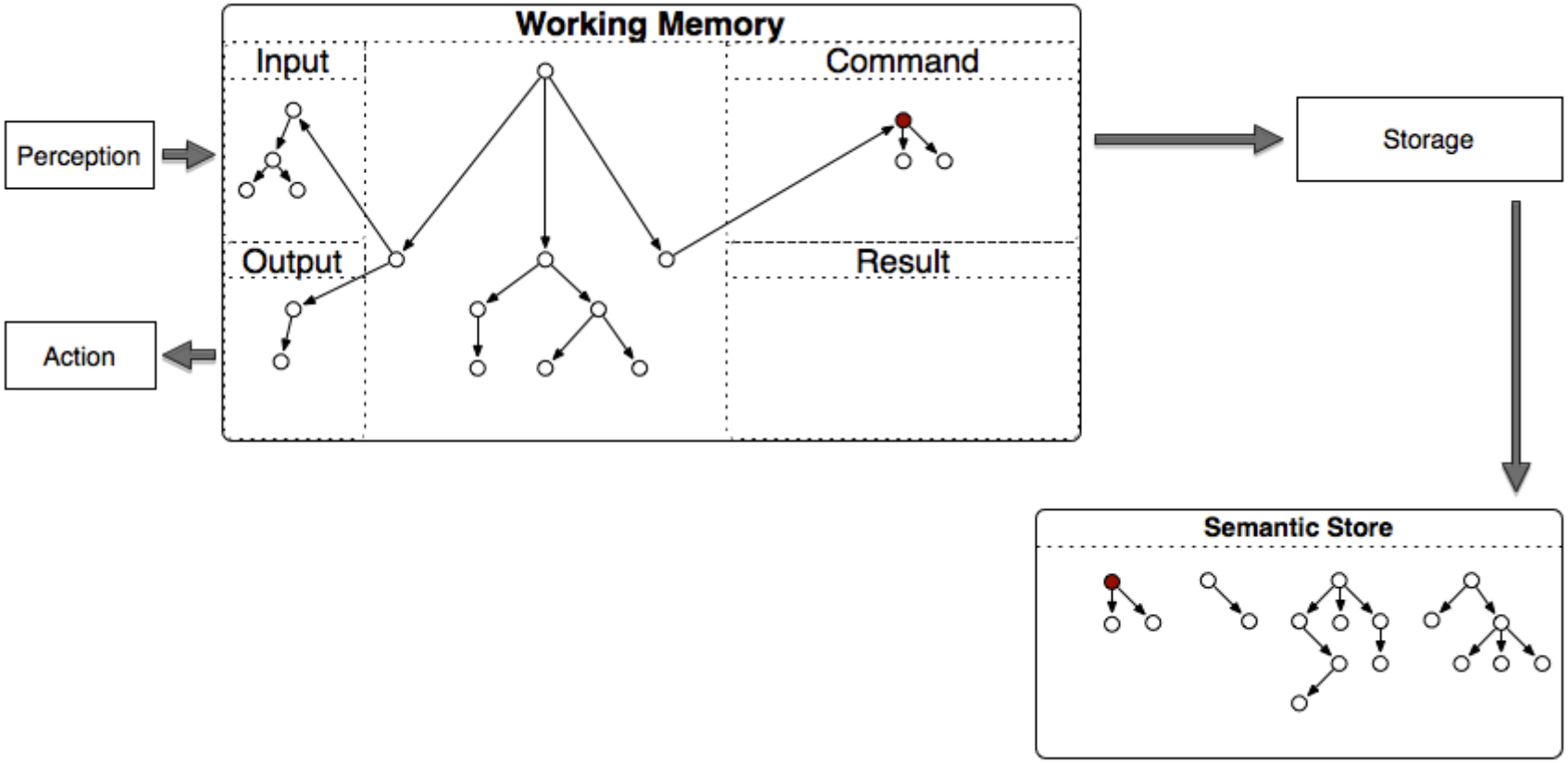
# Architectural Integration

## *Deliberate Storage*



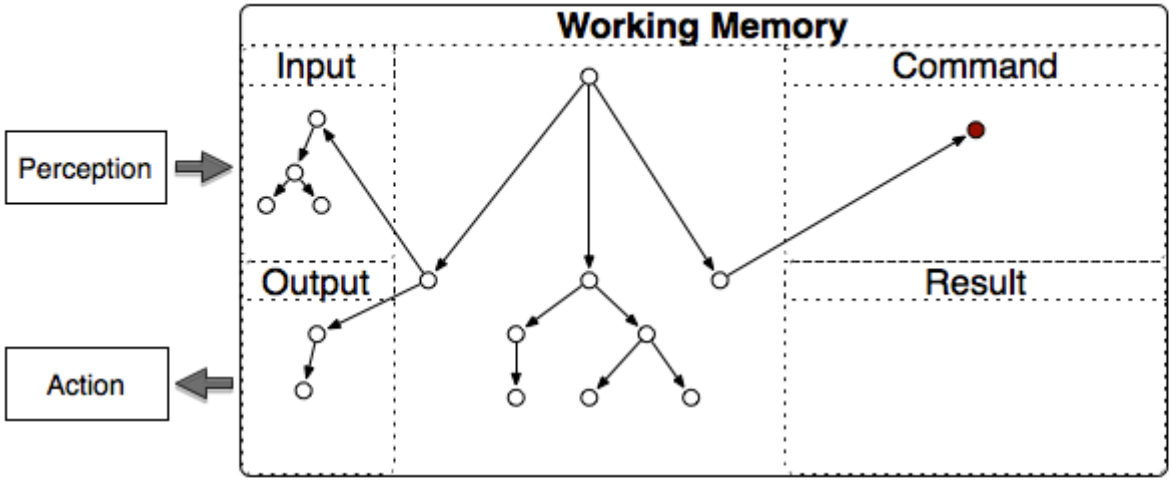
# Architectural Integration

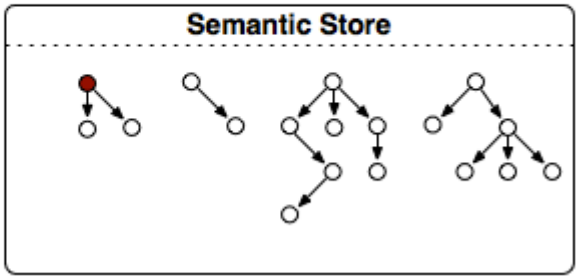
## *Deliberate Storage*



# Architectural Integration

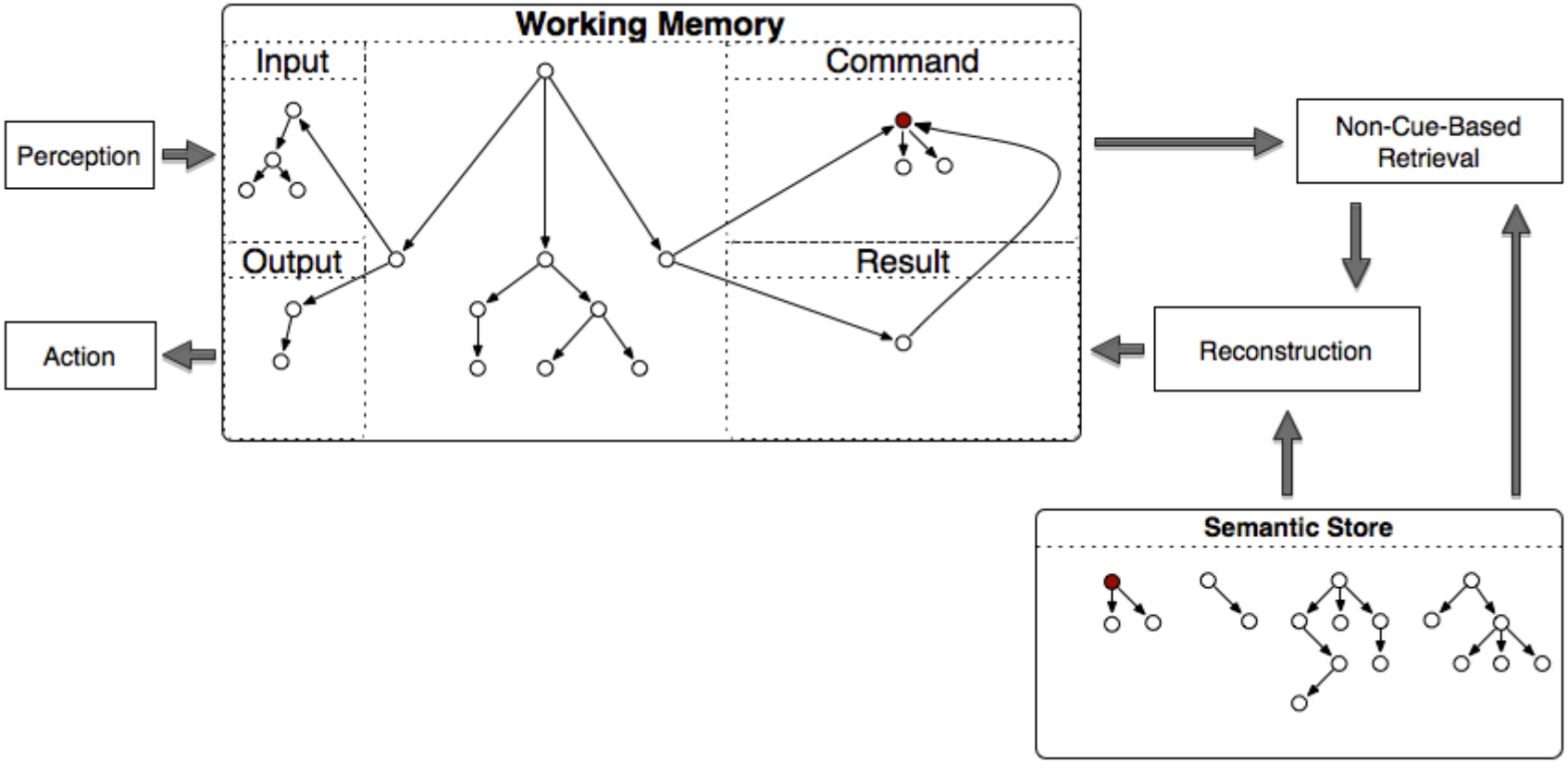
## *Non-Cue-Based Retrieval*



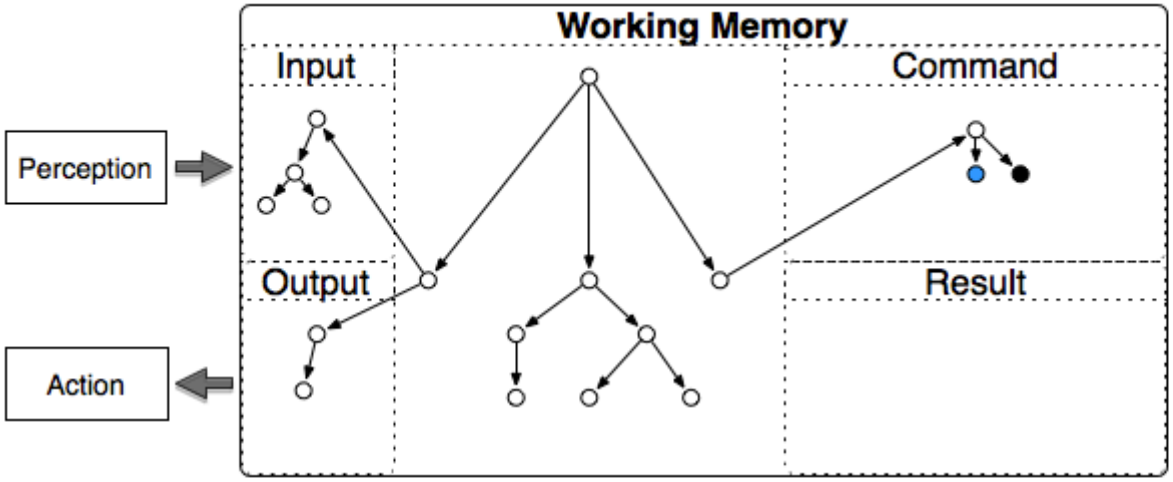
# Architectural Integration

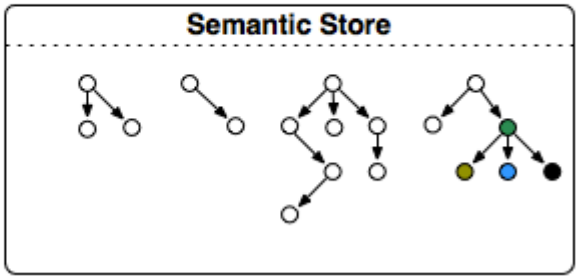
## *Non-Cue-Based Retrieval*



# Architectural Integration

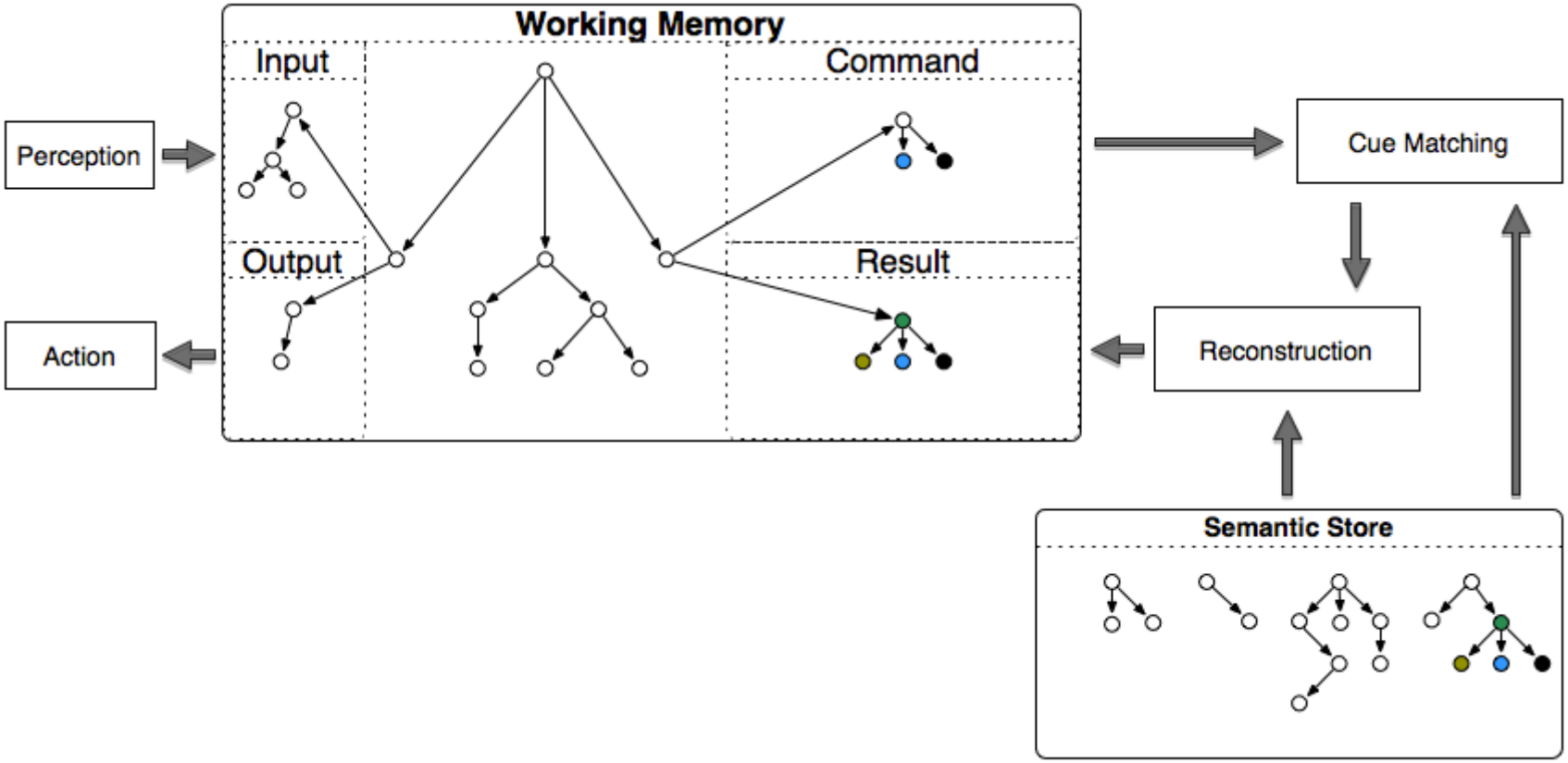
## *Cue-Based Retrieval*



# Architectural Integration

## *Cue-Based Retrieval*



# Basic Usage

- Working-memory structure
- Semantic-memory representation
- Storing knowledge
- Retrieving knowledge

# Working-Memory Structure

Soar creates an `smem` structure on each state

Each `smem` structure has specialized substructure

- `command`: agent-initiated actions
- `result`: architectural feedback

# Semantic-Memory Representation

Similar to working memory: symbolic triples

- All identifiers in semantic memory are *long-term*
  - The letter-number pair (e.g. S5 or C7) is permanently associated with the identifier
  - When printed, long-term identifiers are prefaced with the @ symbol (e.g. @S5 or @C7)
  - When depicted, long-term identifiers are double circles
- Attributes cannot be identifiers (currently)
- The resulting graph is not necessarily connected

# Storing Knowledge

## Manual

Method of appending via command line (especially useful for loading external KBs)

## Agent

Deliberate (via rules) addition/modification

# Manual Storage

Syntax: similar to production RHS

```
smem --add {
   (<id1> ^attr1 val1 val2 ^attr2 val1 … )
   (<id2> ^attr3 <id1> val5 … )
   (<id3> ^attr4.attr5 <id3>)
   …
}
```

# Agent Storage

Syntax

```
(<smem> ^command <cmd>)
(<cmd> ^store <id1> <id2> …)
```

- Processed at end of phase in which rule fires
- Multiple identifiers may be stored at once, but **not** recursive

Result

```
(<smem> ^command <cmd> ^result <r>)
(<cmd> ^store <id1> <id2> …)
(<r> ^success <id1> <id2> …)
```

# Retrieving Knowledge

Non-Cue-Based

Add the features/relations of a known long-term identifier to working memory

Cue-Based

Find a long-term identifier that has a set of features/relations and add it to working memory with its full feature/relation set

Common Constraints (motivated by performance/reactivity):

- Only one per state per decision
  - Processed during *output* phase
- Only re-processed if WM changes to commands

# Non-Cue-Based Retrieval

Syntax

```
(<smem> ^command <cmd>)
(<cmd> ^retrieve <long-term identifier>)
```

Result

```
(<smem> ^command <cmd> ^result <r>)
(<cmd> ^retrieve <long-term identifier>)
(<r> ^<status> <long-term identifier>
     ^retrieved <long-term identifier>)
```

Where `<status>` is…

- `failure`: `<long-term identifier>` is not long-term
- `success`: else (adds all features/relations to WM)

# Cue-Based Retrieval: Syntax

```
(<smem> ^command <cmd>)
(<cmd> ^query <q>)
(<q> ^attr1 val1
     ^attr2 <val2>
     ^attr3 @V3 …)
```

The augmentations of the *query* form hard constraint(s), based upon the value type…

- Constant: exact match
- Long-Term ID: exact match
- Short-Term ID: wildcard

# Cue-Based Retrieval: Result

```
(<smem> ^command <cmd> ^result <r>)
(<cmd> ^query <q>)
(<r> ^<status> <q>
     ^retrieved <long-term identifier>)
```

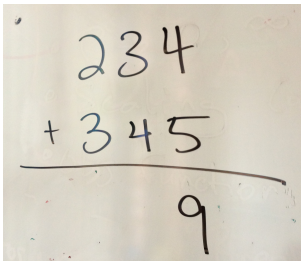
Where `<status>` is…

- `failure`: no long-term identifier satisfies the constraints
- `success`: else (adds all features/relations to WM)

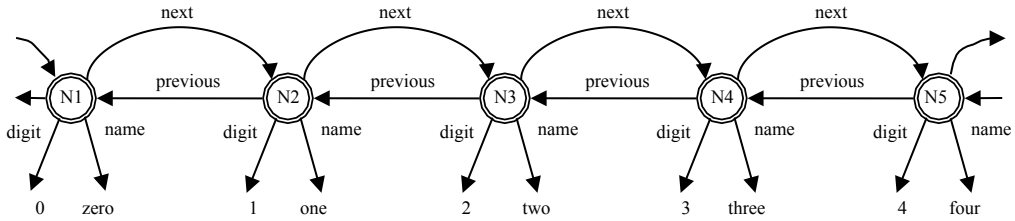
Ties are broken by a bias

- Default = recency, also supports frequency & base-level activation (more on this shortly)

# Example Task
## *Multi-Column Arithmetic*



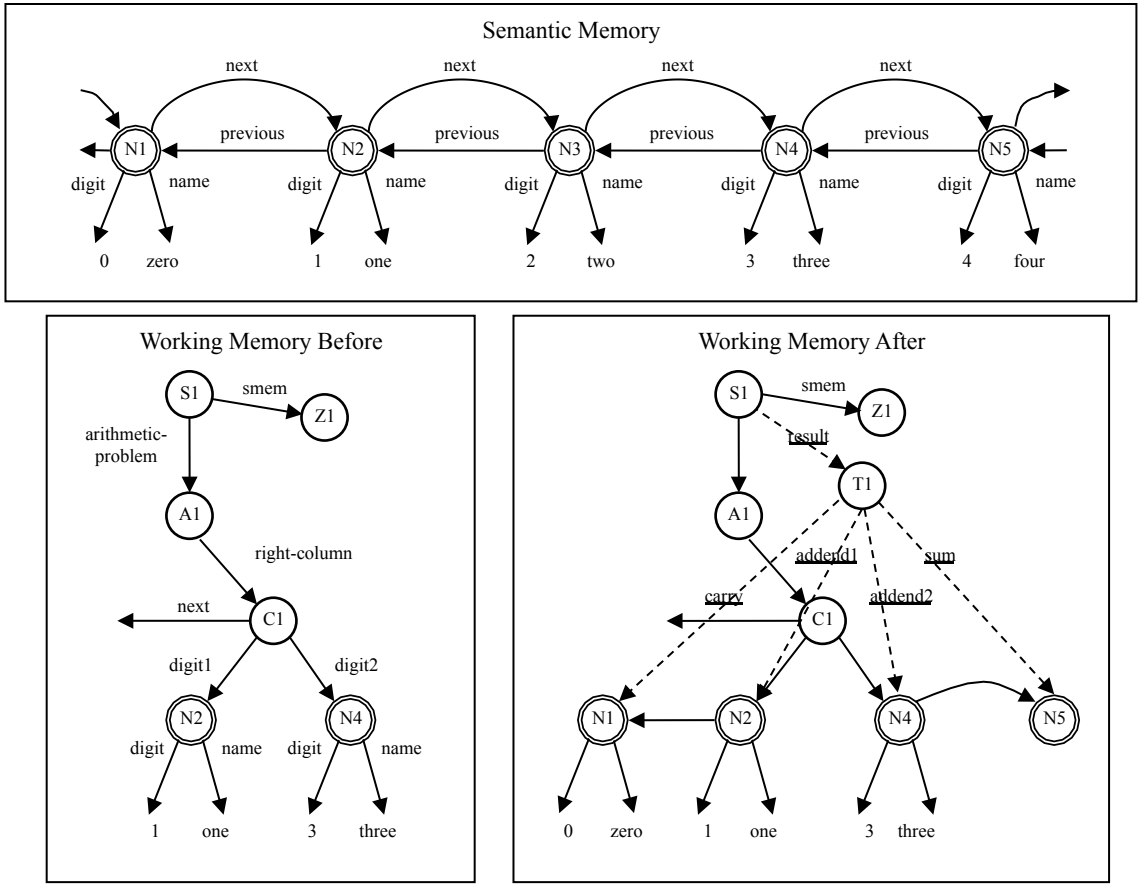

- Given, **SMem**: 0 <-> 1 <-> 2 … 8 <-> 9




- Given, **Rules**: process a column in a substate
  - Input: 4+5, Output: sum=9, carry=0
  - Automatically chunks result, given exact context
  - Deliberately store to SMem
    - Independent of context, so can flexibly query… 5+4=? Or 9=5+? (i.e. subtraction)

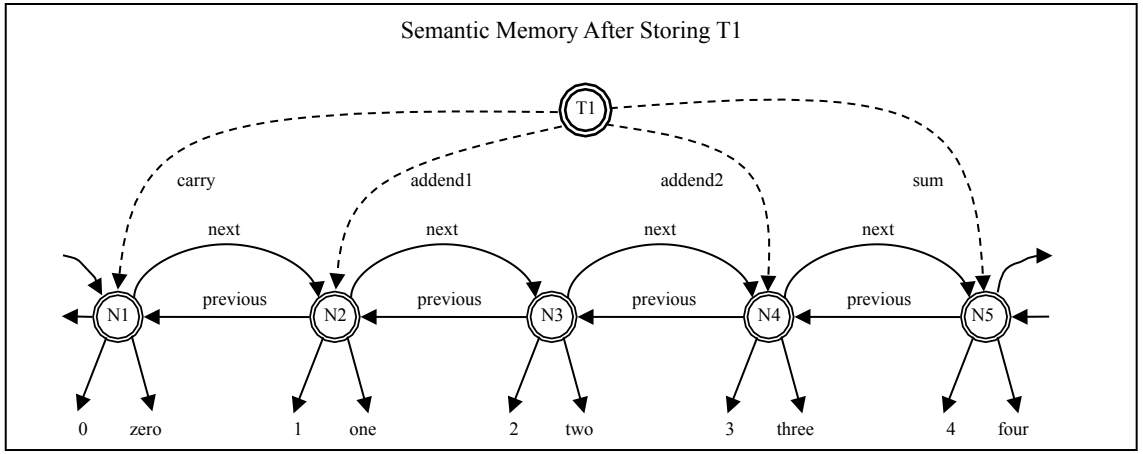
# Multi-Column Arithmetic

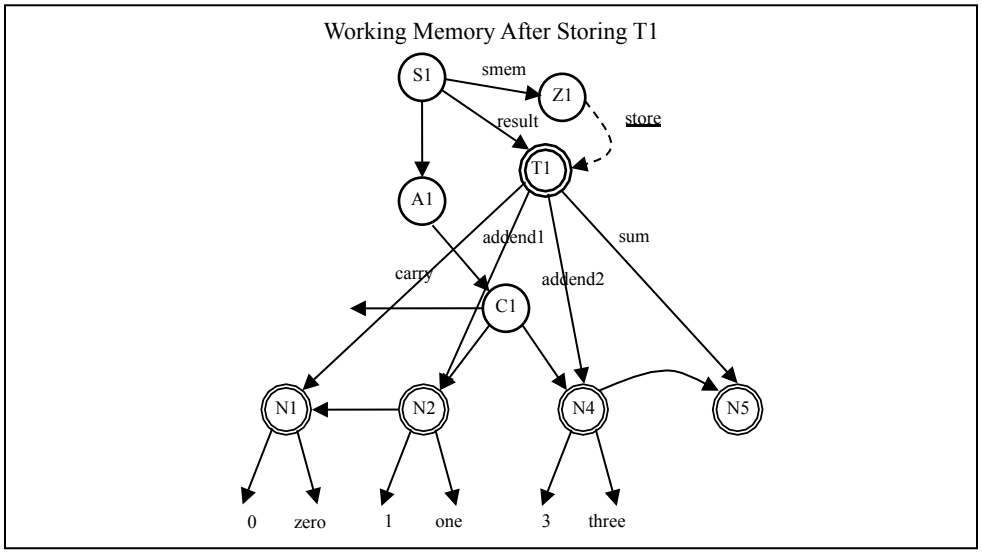
## *Substate Calculation via Rules*



# Multi-Column Arithmetic

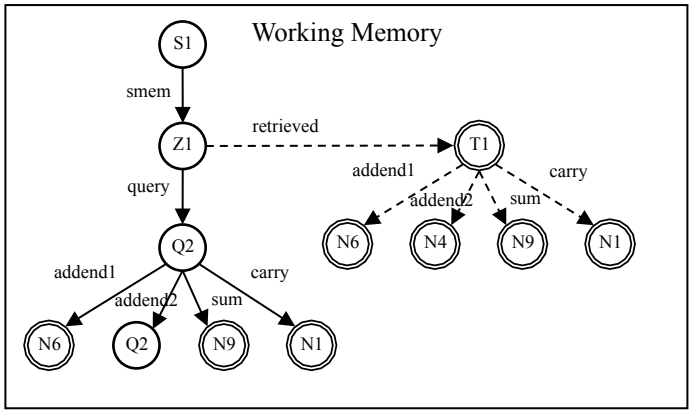
## *SMem Storage*



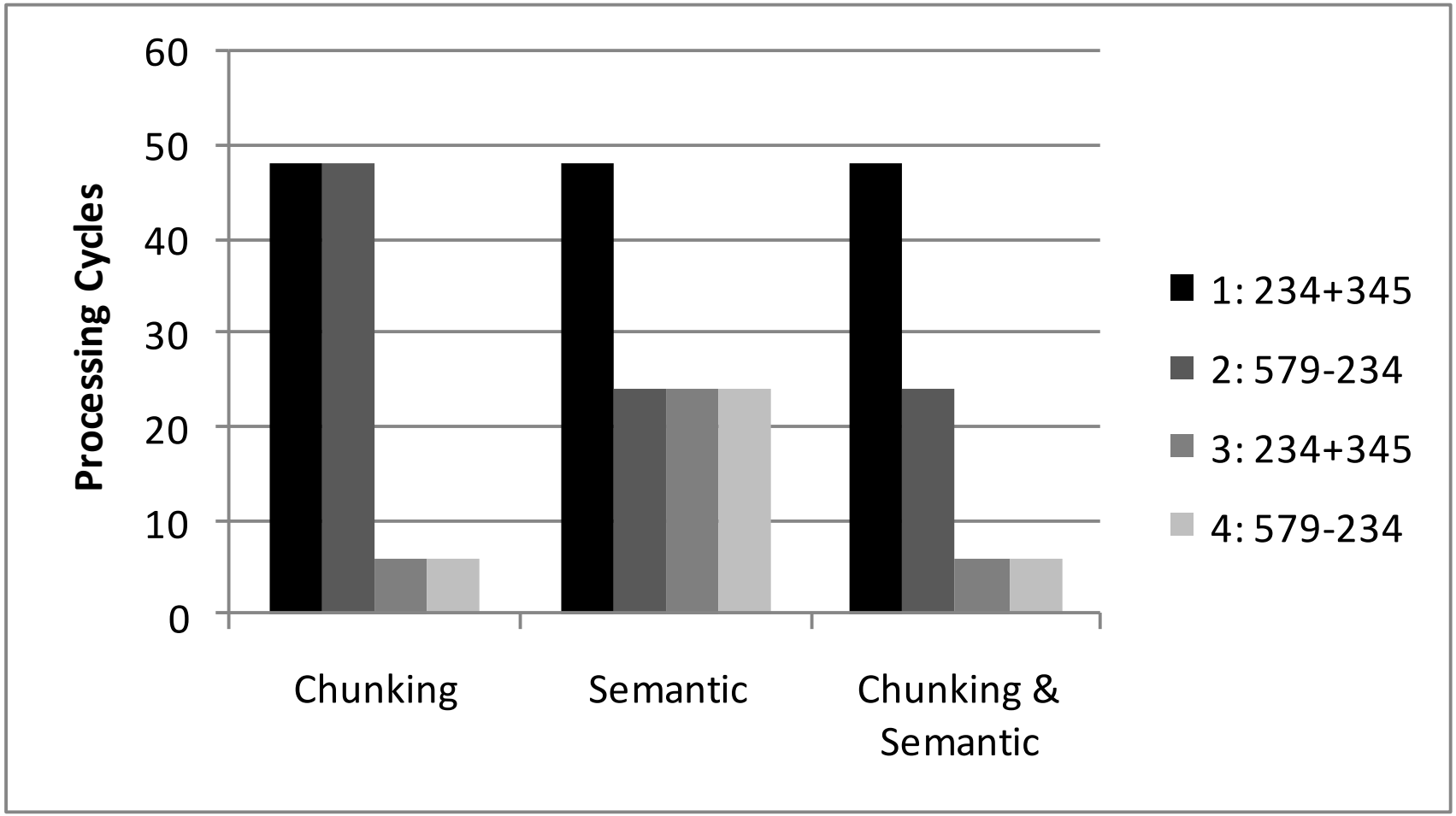
# Multi-Column Arithmetic

## *SMem Retrieval*



# Multi-Column Arithmetic

## *Procedural Learning Interactions*



# Semantic Memory

## *Computational Challenges for Scaling RT Agents*

Dynamic…

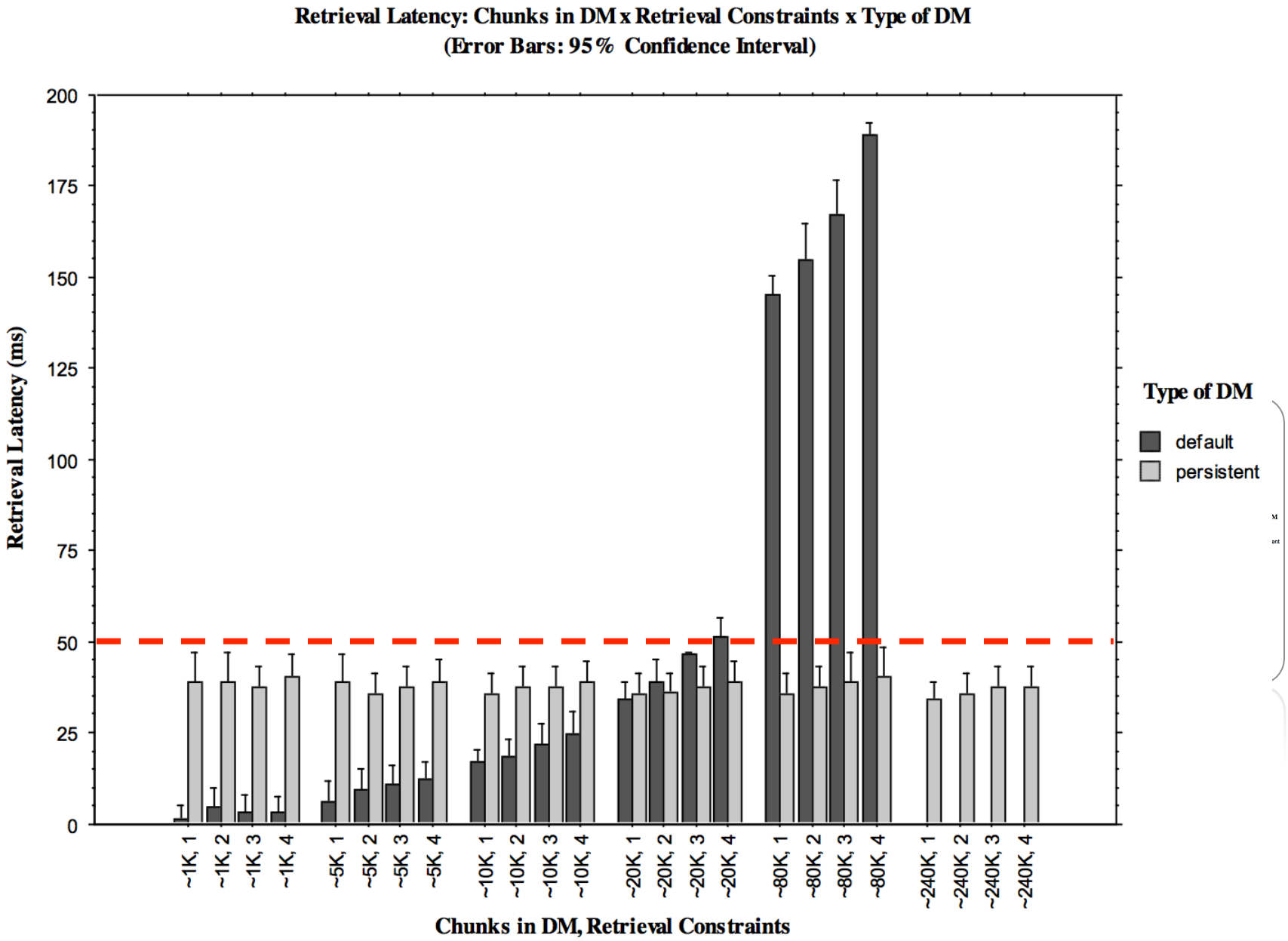
- number of nodes/edges
- symbol vocabulary

Scaling potential

- Nodes ~ millions
- Edges ~ 10 per node

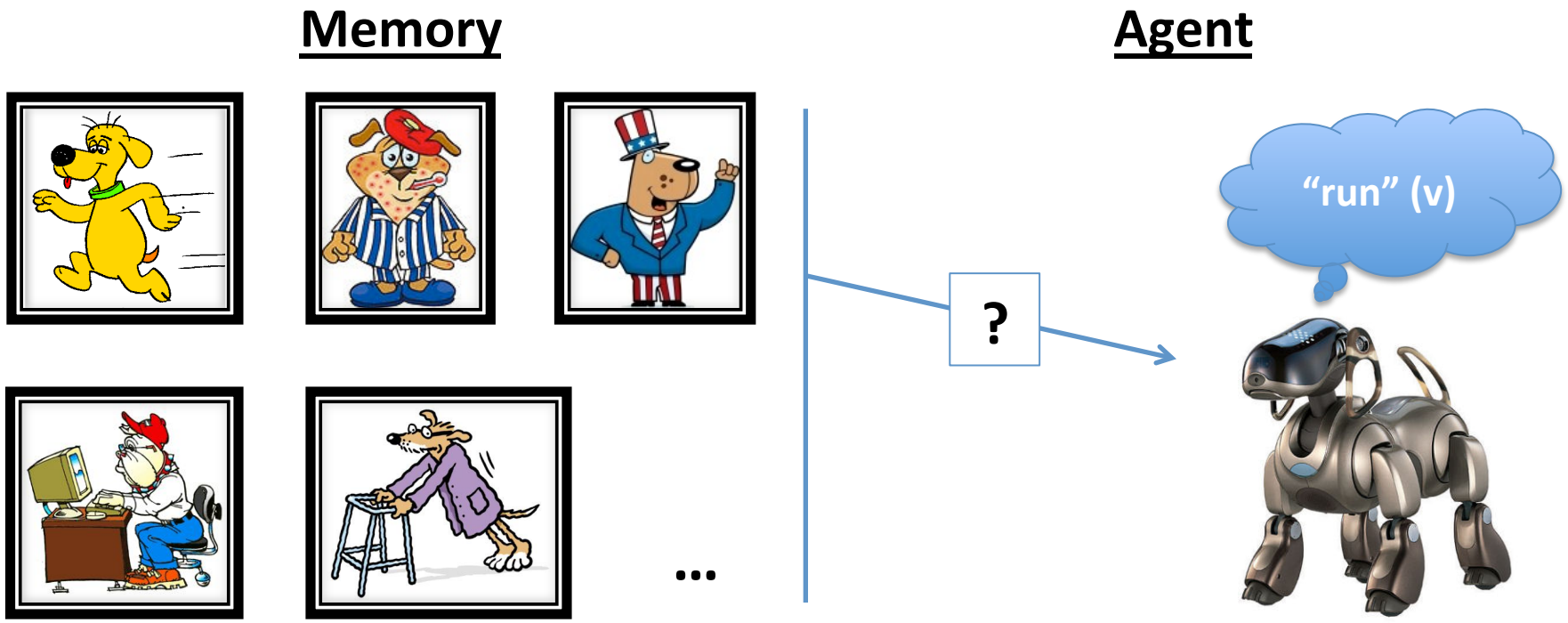
Cue-matching optimality

- Feature satisfaction, ranking w.r.t. bias value
- *O*( |cue| x |objects| )


Retrieval Latency: Chunks in DM x Retrieval Constraints x Type of DM
(Error Bars: 95% Confidence Interval)
Retrieval Latency (ms)
200
175
150
125
100
75
50
25
0
~1K, 1
~1K, 2
~1K, 3
~1K, 4
~5K, 1
~5K, 2
~5K, 3
~5K, 4
~10K, 1
~10K, 2
~10K, 3
~10K, 4
~20K, 1
~20K, 2
~20K, 3
~20K, 4
~80K, 1
~80K, 2
~80K, 3
~80K, 4
~240K, 1
~240K, 2
~240K, 3
~240K, 4
Chunks in DM, Retrieval Constraints
Type of DM
default
persistent

# Evaluation

## *Motivation*




**Problem**. Ambiguous Cues
**Hypothesis**. Retrieval History is Useful
**Application**. Word Sense Disambiguation

# Evaluation
## *Historical Memory Retrieval Bias*

Experimental Setup

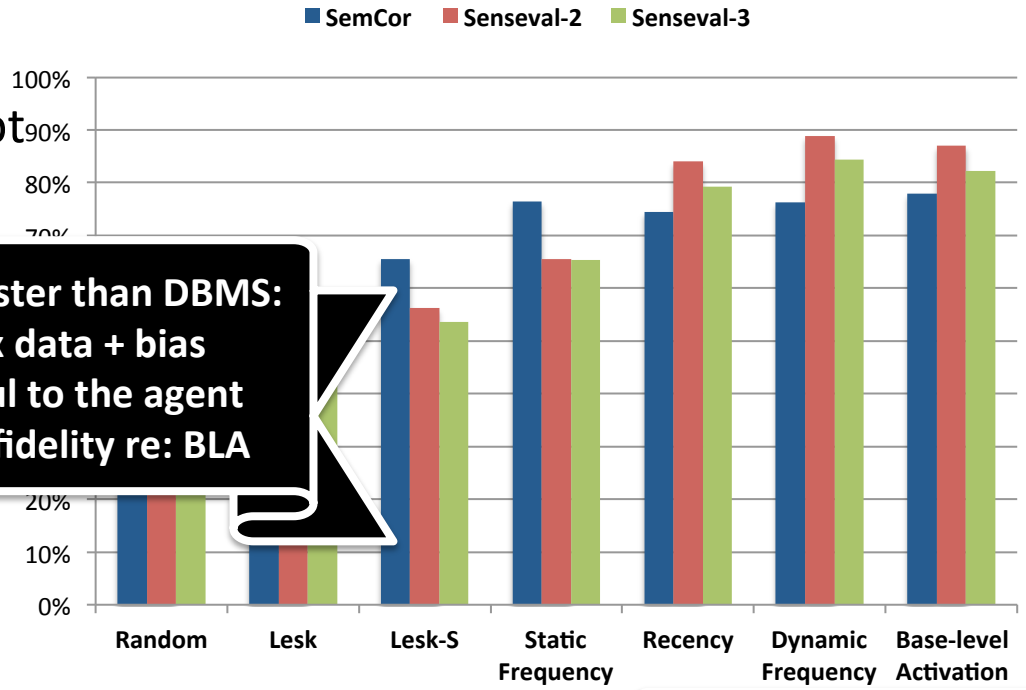
- Input: "word", POS
- Given: WordNet v3
- Correct sense(s) after each attempt

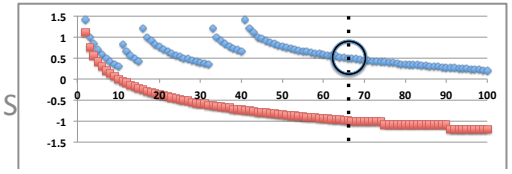
Efficiency & Scaling

- R/DF: *O*(1), ≤ 0.87 msec.
- Base-Level Activation:
  - Naïve: *O*(# cand's), ≤ 13.
  - *Locally Efficient* Appro: *O*(1), ≤ 1.34 msec.
    *Idea: relative ranking is all that is required, so re-compute {k[th]} recent memory as a heuristic (>90% fidelity)*

**>30x faster than DBMS:**
**>3x data + bias**
**Useful to the agent**
**High-fidelity re: BLA**

Task Performance (**2nd corpus exp.**)




**Biased Retrievals**

# Comparison to DM in ACT-R

Storage

- Deliberate action by the agent, no merging
- Representation supports multi-valued attributes (e.g. ^feature a ^feature b)
- No concept of chunk "types" – attributes are fully dynamic

Retrieval

- Does not support noise, activation threshold, negated queries, partial match, spreading
- Uses SQLite, inverted indexes, and heuristic search to scale real-time performance to very large data sets (GBs ~ millions of nodes/edges); more on this later

# Thank You :)

**Questions?**