

# Episodic-Memory Tutorial

Soar Workshop 33 – Nate Derbinsky

While waiting...

1. Make sure you have internet access

2. Download Soar Tutorial package

[web.eecs.umich.edu/~soar/workshop\\_tutorial](http://web.eecs.umich.edu/~soar/workshop_tutorial)

3. Download Graphviz

[www.graphviz.org](http://www.graphviz.org)

4. Download Eclipse (with at least Java)

[www.eclipse.org](http://www.eclipse.org)

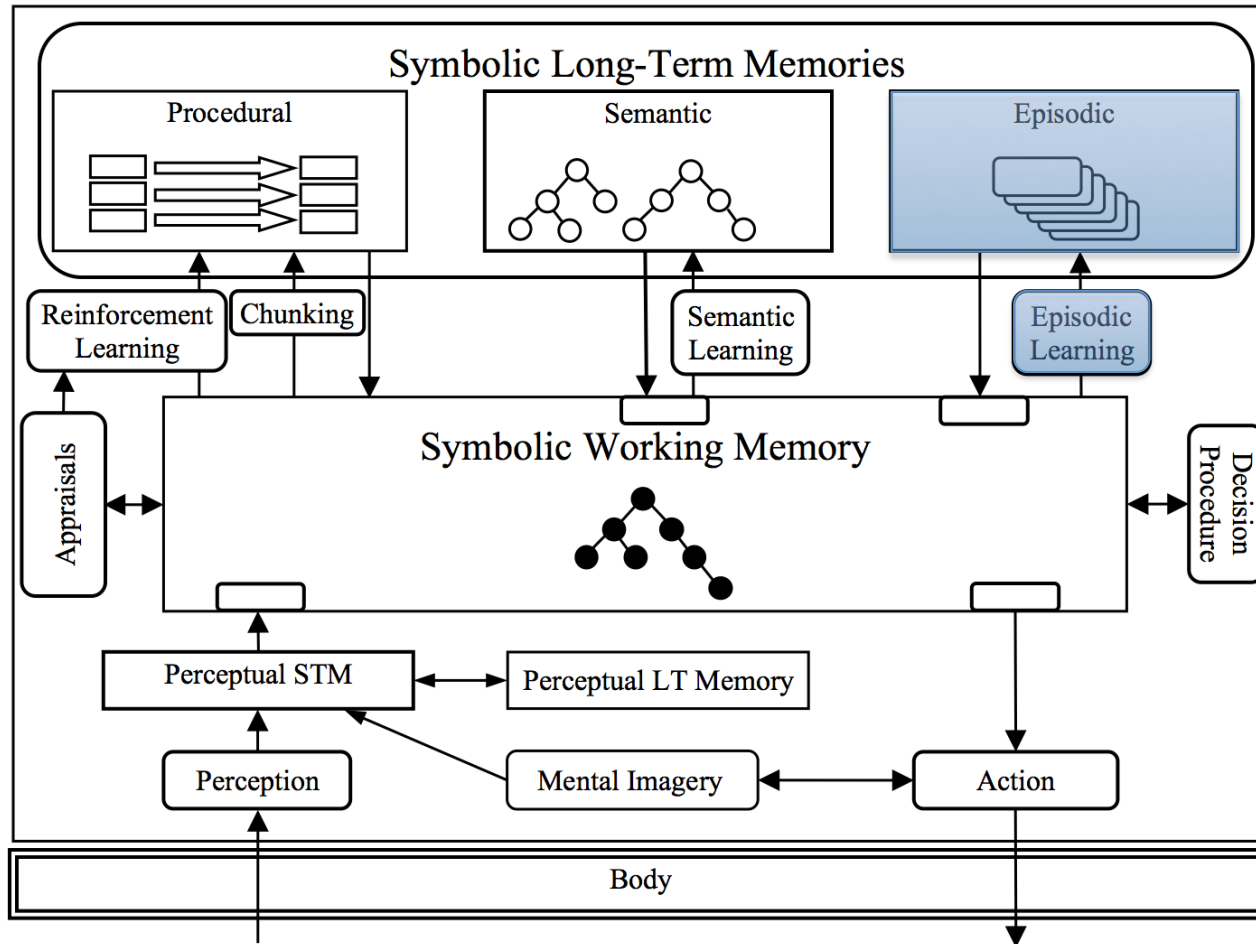
5. Download tutorial support files

[web.eecs.umich.edu/~nlderbin/workshop33](http://web.eecs.umich.edu/~nlderbin/workshop33)

# Agenda

- Big picture
- Basic usage
- Demo task
- Additional resources

# Soar 9

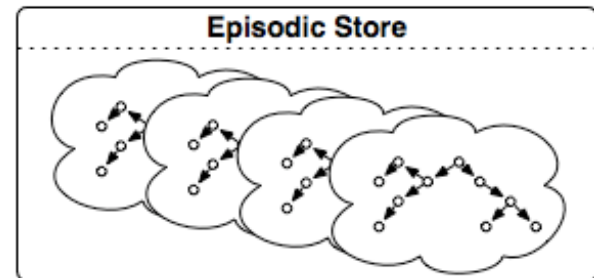
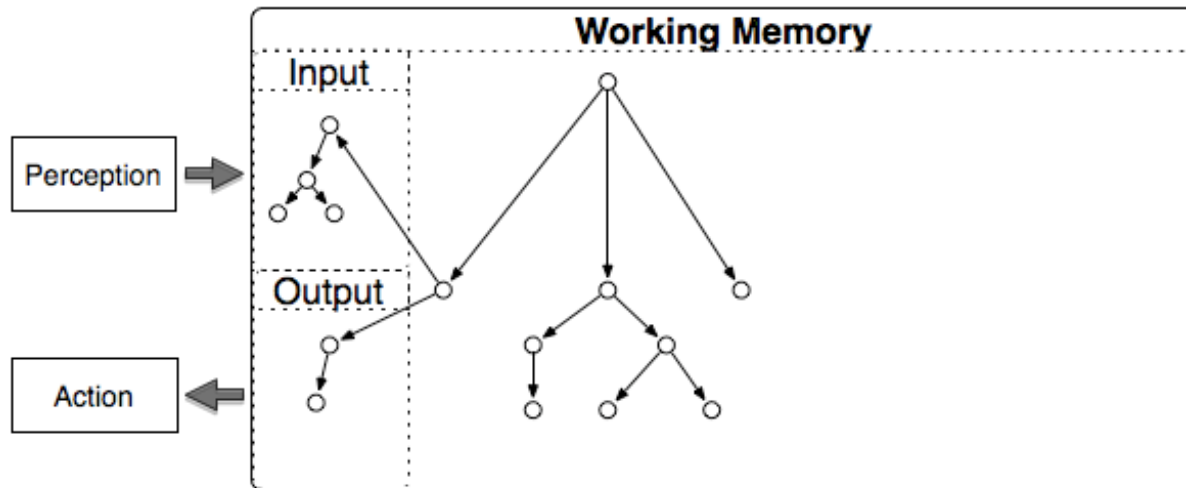


# Episodic Memory: Big Picture

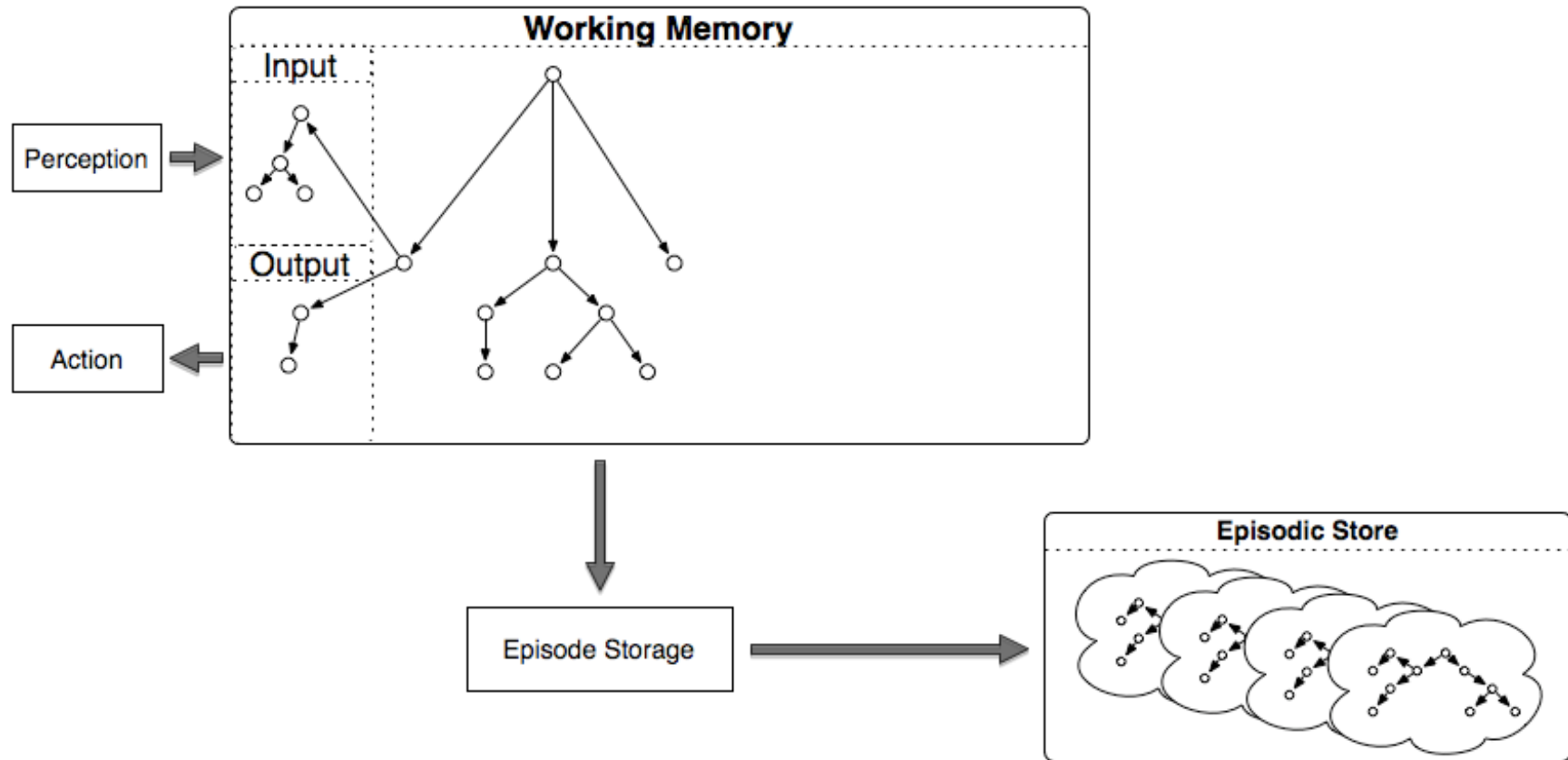
Episodic memory is a weak learning mechanism

- Automatically captures, stores, and temporally indexes agent state
- Supports content-addressable agent interface to autobiographical prior experience

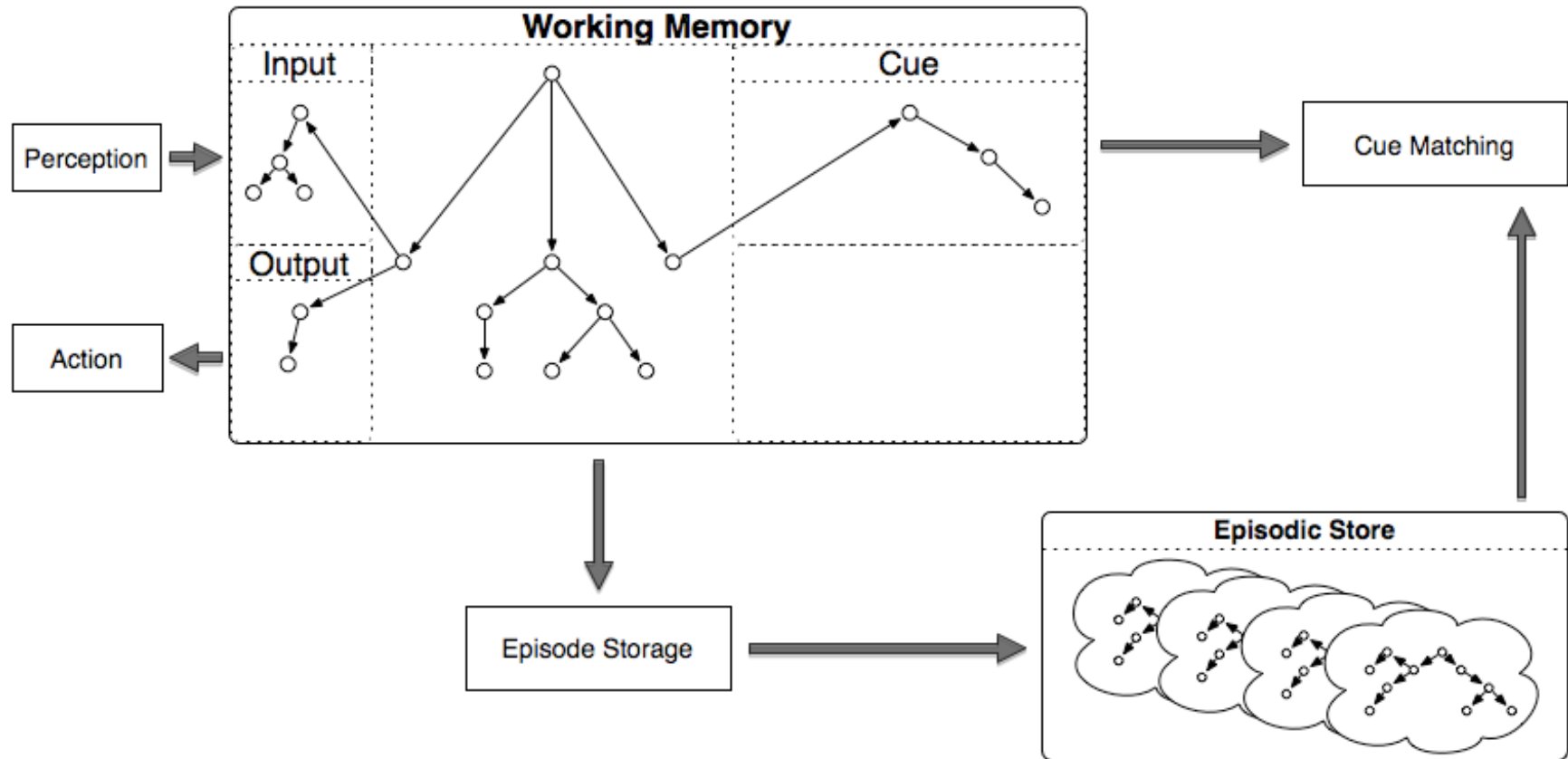
# Architectural Integration



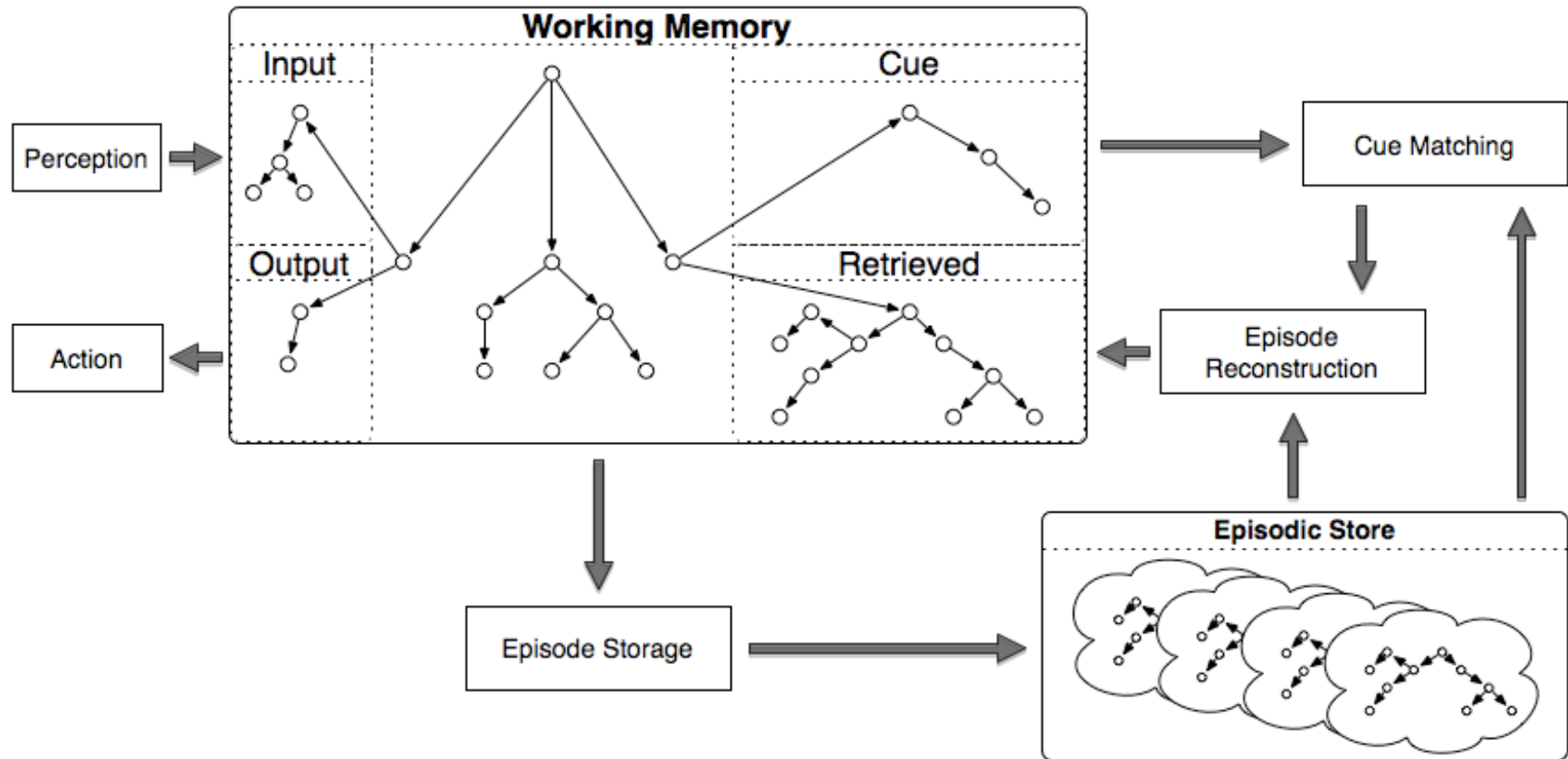
# Architectural Integration



# Architectural Integration



# Architectural Integration





# Basic Usage

- Working-memory structure
- Episodic-memory representation
- Controlling episodic memory
- Storing knowledge
- Retrieving knowledge

# Working-Memory Structure

Soar creates an `epmem` structure on each state

- Soar Java Debugger
  - `step 5`
  - `print --exact (* ^epmem *)`
  - `print e1`

Each `epmem` structure has specialized substructure

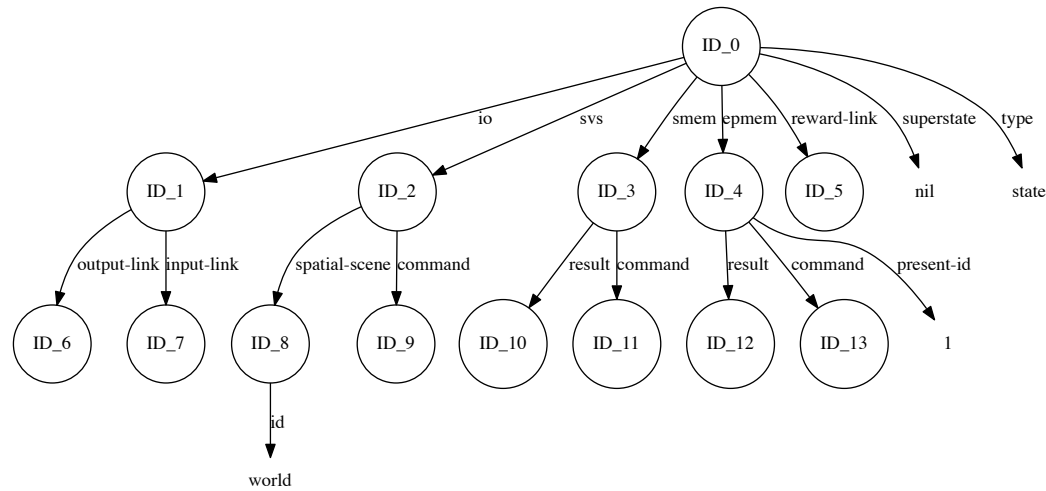
- `command`: agent-initiated actions
- `result`: architectural feedback
- `present-id`: current episode number (more later)

# Episodic-Memory Representation

Similar to working memory: symbolic triples

- Attributes cannot be identifiers (currently)
- Structures within an episode are connected; separate episodes are disconnected

```
(<id0> ^epmem <id4>
^io <id1>
^reward-link <id5>
^smem <id3>
^superstate nil
^svs <id2>
^type state)
(<id1> ^input-link <id7>
^output-link <id6>)
(<id2> ^command <id9>
^spatial-scene <id8>)
(<id3> ^command <id11> ^result <id10>)
(<id4> ^command <id13> ^present-id 1 ^result <id12>)
(<id8> ^id world)
```



# Controlling Episodic Memory

Get/Set a parameter:

- `epmem [-g|--get] <name>`
- `epmem [-s|--set] <name> <value>`

EpMem is **disabled** by default. To enable it...

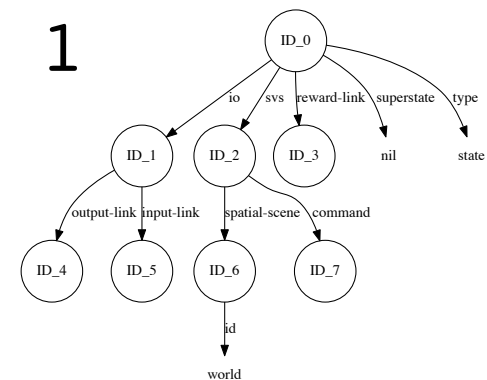
1. `epmem`
2. `epmem --set learning on`
3. `epmem`

# Storing Knowledge

- Automatic storage requires EpMem to be **enabled** (see slide 12)
- Storage captures the top state of working memory
- Events trigger storage of new episodes
  - `epmem --set trigger << dc output >>`
    - `dc`: decision cycle (default – **change since v9.3.2**)
    - `output`: new augmentation of output-link
- Storage takes place at the end of a phase
  - `epmem --set phase << output selection >>`
    - `output` is default
    - `selection` may be useful for in-the-head agents

# Automatic Storage: Example

- Soar Java Debugger
  1. `epmem --set learning on`
  2. `watch --epmem`
  3. `run 5 -p`
  4. `epmem --print 1`
  5. `ctf ep.gv epmem --viz 1`
  6. `print e1`
  7. `epmem --stats`



# Automatic Storage: Debrief

- What wasn't captured?
- Attributes can be excluded from encoding (and subsequent recursion)
  - `epmem --set exclusions <label>`
    - If `<label>` already excluded, now included
- Try previous example, add before #1:
  - `epmem --set exclusions epmem`
  - `epmem --set exclusions smem`

# Retrieving Knowledge

## Cue-Based

Find the episode that best matches a cue and add it to working memory

## Temporal Progression

Replace the currently retrieved episode with the next/previously encoded episode

## Non-Cue-Based (not covered)

Add an episode to working memory from episode #

## Common Constraints:

- Requires that EpMem is enabled (slide 12)
- Only one per state per decision
- Processed during phase (slide 13)
- Only re-processed if WM changes to commands
- Meta-data (status, etc) automatically cleaned by the architecture



# Cue-Based Retrieval: Syntax

( <epmem> ^command <cmd> )  
( <cmd> ^query <q>  
          ^neg-query <nq> )

- The `neg-query` is optional
- Cues must be acyclic
- The `<q>` and `<nq>` identifiers form the roots of episode sub-graph cues
  - `query` represents desired structures
  - `neg-query` represents undesired structures

# Cue-Based Retrieval: Cue Semantics

Values of cue WMEs are interpreted by type

- Constant: exact match
- Long-Term ID: exact match, stop
- Short-Term ID: Wildcard (but must be identifier)

# Cue-Based Retrieval: Episode Scoring

- **Leaf WME**, either...
  - Cue WME whose value is a constant/long-term identifier OR
  - Cue WME whose value is a short-term identifier and that identifier has no augmentations
- A leaf wme is *satisfied* (w.r.t. an episode) if...
  - The episode contains that WME AND
  - The episode contains a path from root to that WME
- Episode scoring
  - $(\text{balance})(\text{cardinality}) + (1-\text{balance})(\text{activation})$
  - balance: parameter=[0,1], default=1
  - cardinality: # satisfied leaf WMEs
  - activation:  $\Sigma$  satisfied leaf WME activation (see Manual)
  - cardinality/activation negated for neg-query

# Cue-Based Retrieval: Cue Matching

## Graph matching

```
epmem --set graph-match << on off >>
```

- on by default

## Candidate episode

Defined as satisfying at least one leaf WME

Cue matching will return the most recent graph-matched episode, or the most recent non-graph-matched candidate episode with the maximal episode score

# Cue-Based Retrieval: Result

Augmentation	Meaning
<code>^retrieved &lt;retrieval-root&gt;</code>	Root of the retrieved memory
<code>^&lt;&lt; success failure &gt;&gt; &lt;query&gt; &lt;neg-query&gt;</code>	Query status
<code>^match-score #</code>	Float, episode score (slide 19)
<code>^cue-size #</code>	Integer, number of leaf WMEs
<code>^normalized-match-score #</code>	match-score/cue-size
<code>^match-cardinality #</code>	Integer, number of satisfied leaf WMEs ( $ \text{query}  -  \text{neg-query} $ )
<code>^memory-id #</code>	Integer, episode # retrieved
<code>^present-id #</code>	Integer, current episode #
<code>^graph-match &lt;&lt; 0 1 &gt;&gt;</code>	Integer, 1 if graph match succeeded
<code>^mapping &lt;mapping-root&gt;</code>	A mapping from the cue to episode

# Cue-Based Retrieval: Example

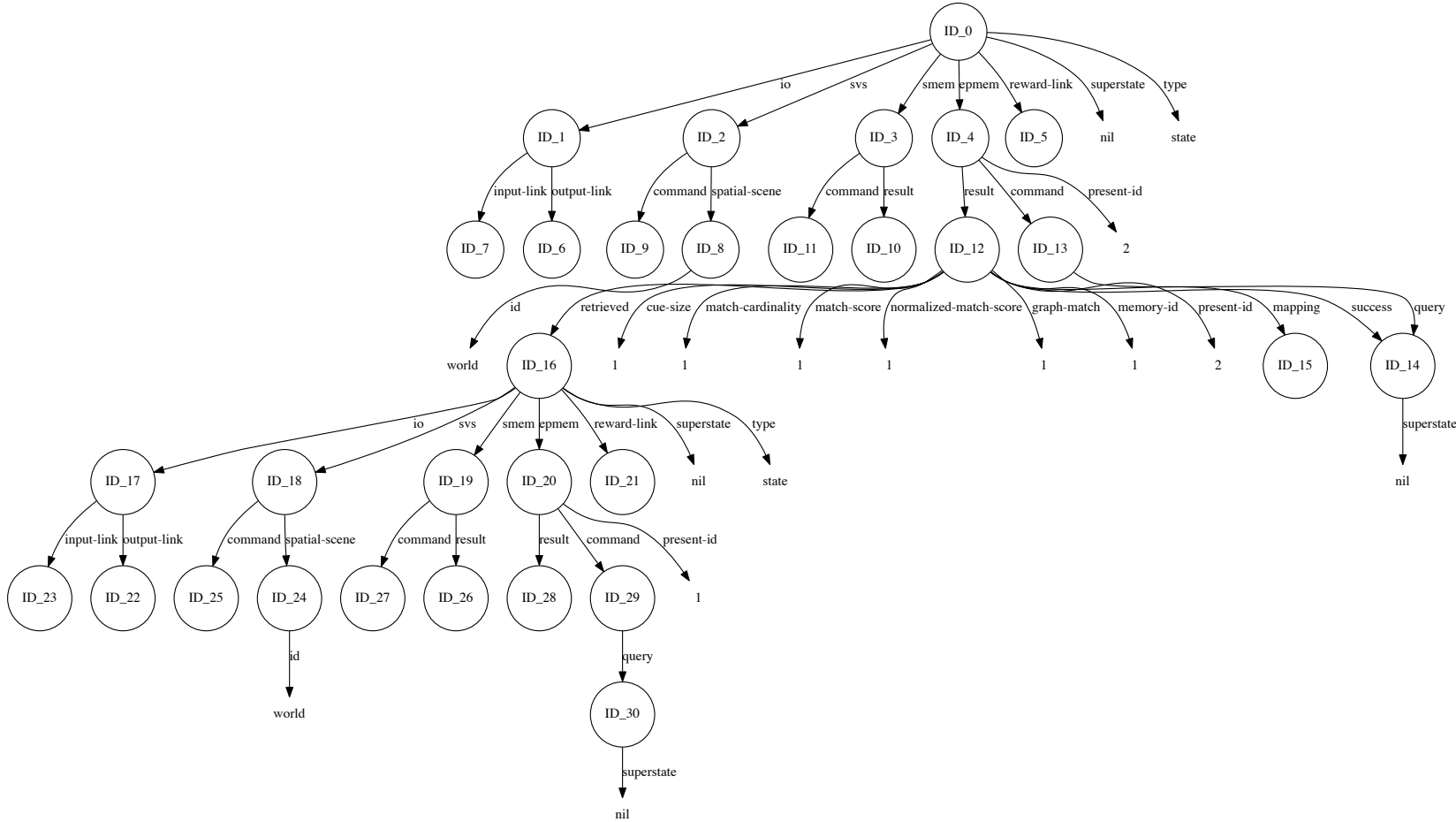
*query-1.soar (find ^superstate nil)*

- Soar Java Debugger

1. `epmem --set learning on`
2. `watch --epmem`
3. `sp {query1  
    (state <s> ^superstate nil  
        ^epmem.command <cmd>)  
    -->  
    (<cmd> ^query.superstate nil)}`
5. `run 5 -p`
6. `print -d 10 e1`

# Cue-Based Retrieval: Example

## Result



# Cue-Based Retrieval: Example

## *Trace*

```
CONSIDERING EPISODE (time, cardinality, score): (1, 1, 1.000000)  
NEW KING (perfect, graph-match): (true, true)
```



# Cue-Based Retrieval

## *Optional Modifiers*

(`<cmd> ^before time-id`)

(`<cmd> ^after time-id`)

(`<cmd> ^prohibit time-id1 time-id2 ...`)

Hard constraints on the episodes that can be retrieved.

# Temporal Progression

(`<cmd> ^next <new-id>`)

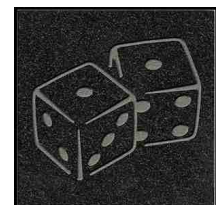
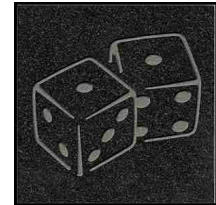
(`<cmd> ^previous <new-id>`)

Retrieves the next/previous episode, temporally, with respect to the last that was retrieved

# Demo Task: Virtual Sensing

*demo.soar*

1. Produce a random number in WM  
EpMem automatically records this episode  
(`demo-start.soar`)
2. Remove the number from WM  
Write to the trace (for later verification)
3. Query episodic memory  
When did I last see a random number?
4. Reason about the retrieved episode  
Extract and print the number



# Additional Resources

- Documentation
- Demo agents
- Readings

# Documentation

Manual and Tutorial  
Documentation/

## Additional Topics

- Absolute non-cue-based retrievals
- Disk-based databases
- Performance
- Usage: commands, parameters, statistics, etc.
- ...

# Demo Agents

[code.google.com/p/soar/wiki/DLCategory\\_Agents](http://code.google.com/p/soar/wiki/DLCategory_Agents)

- kb
  - Demonstrates and unit tests the EpMem API
- count-epmem
  - Counting agent: stores then retrieves
  - Used for performance evaluation

# Select Readings

*code.google.com/p/soar/wiki/Publications*

2004

- A Cognitive Model of Episodic Memory Integrated with a General Cognitive Architecture  
Andrew M. Nuxoll, John E. Laird (ICCM)

2007

- Extending Cognitive Architecture with Episodic Memory  
Andrew M. Nuxoll, John E. Laird (AAAI)

2009

- Efficiently Implementing Episodic Memory  
Nate Derbinsky, John E. Laird (ICCBR)
- A Year of Episodic Memory  
John E. Laird, Nate Derbinsky (IJCAI Workshop)

2010

- Extending Soar with Dissociated Symbolic Memories  
Nate Derbinsky, John E. Laird (AISB)
- Instance-Based Online Learning of Deterministic Relational Action Models  
Joseph Xu, John E. Laird (AAAI)

2011

- Learning to Use Episodic Memory  
Nicholas A. Gorski, John E. Laird (Cognitive Systems Research)

2012

- Enhancing Intelligent Agents with Episodic Memory  
Andrew M. Nuxoll, John E. Laird (Cognitive Systems Research)
- A Multi-Domain Evaluation of Scaling in a General Episodic Memory  
Nate Derbinsky, Justin Li, John E. Laird (AAAI)