

Soar-SMem Tutorial

Soar Workshop 31 – Nate Derbinsky

While waiting...

1. Make sure you have internet access

2. Download Soar 9.3.1

soar.googlecode.com

3. Download Graphviz

www.graphviz.org

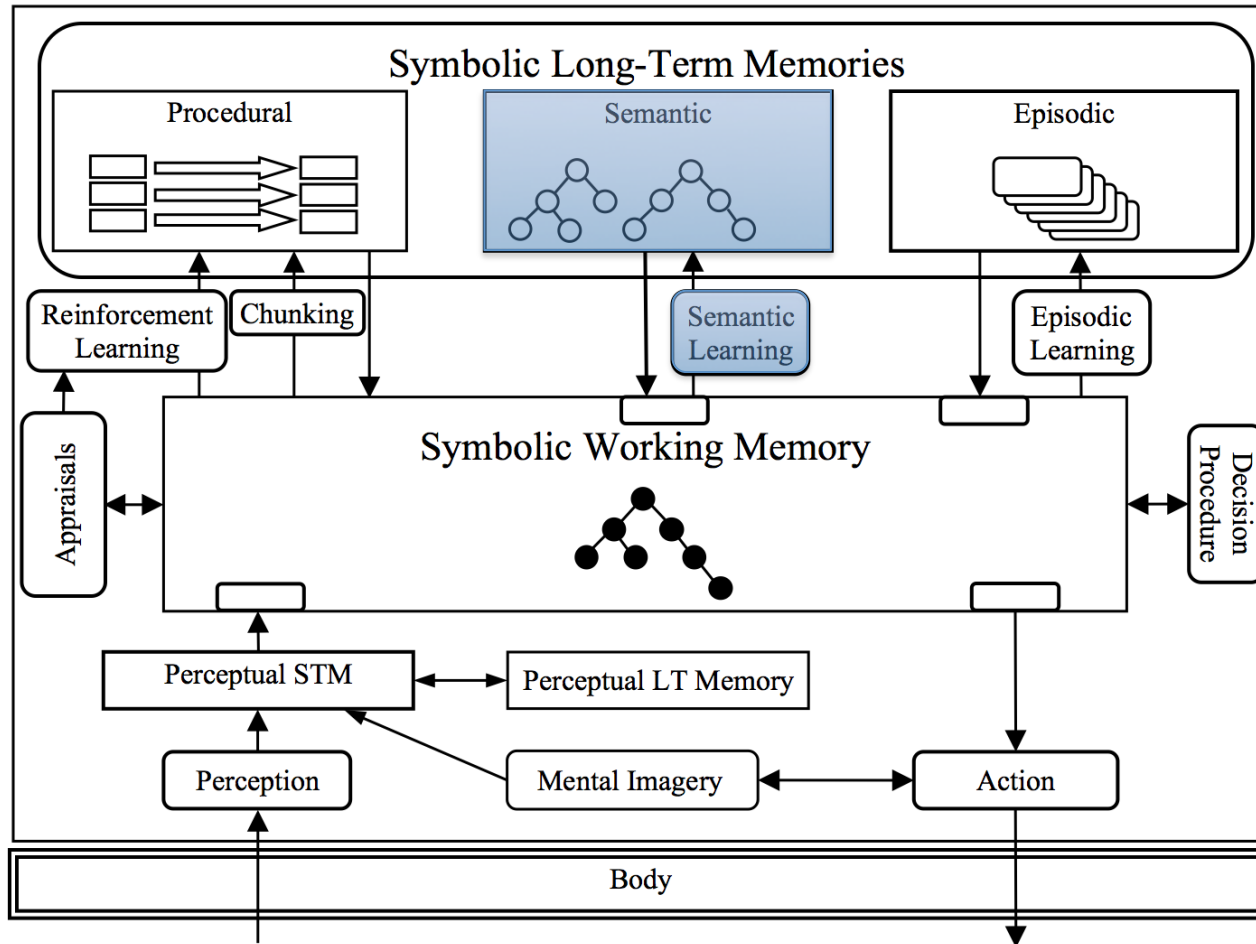
4. Download tutorial support files

www.eecs.umich.edu/~nlderbin/workshop31

Agenda

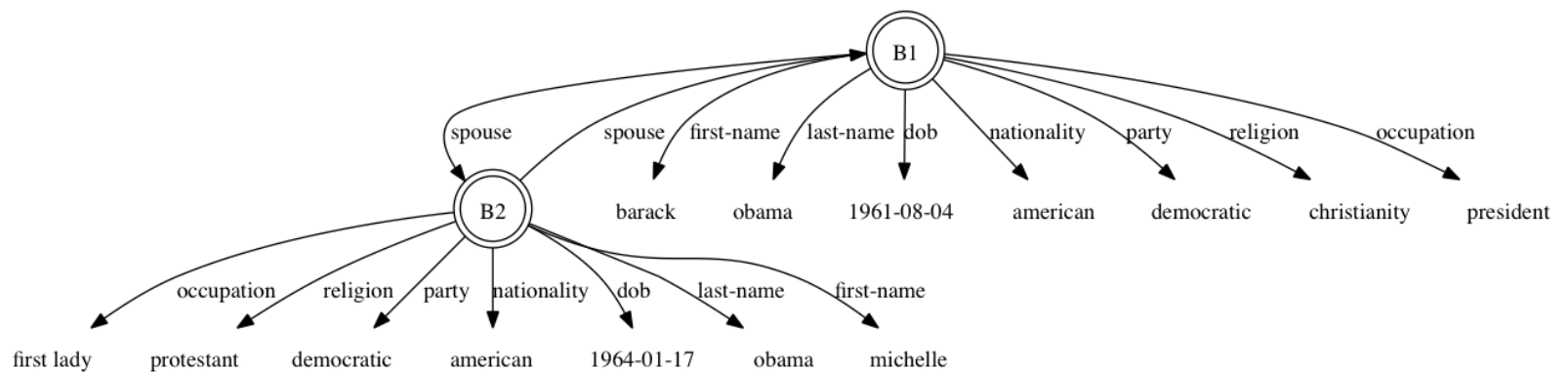
- Big picture
- Basic usage
- WordNet demo
- Additional resources

Soar 9

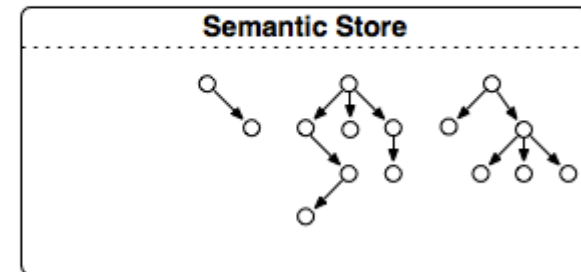
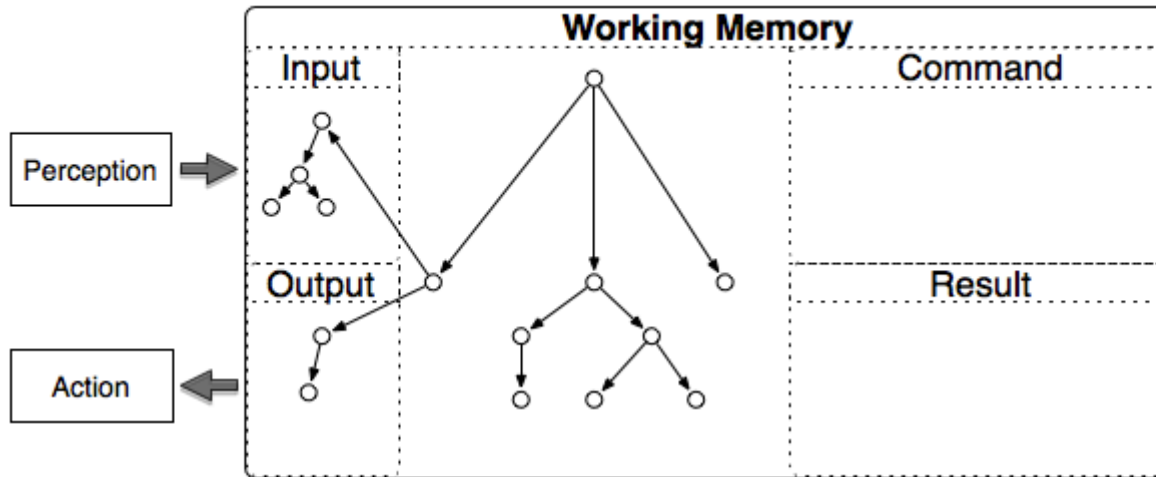


Semantic Memory: Big Picture

Supports deliberate storage and retrieval of long-term objects, features, and relations

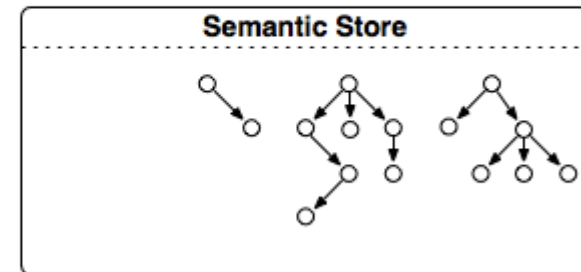
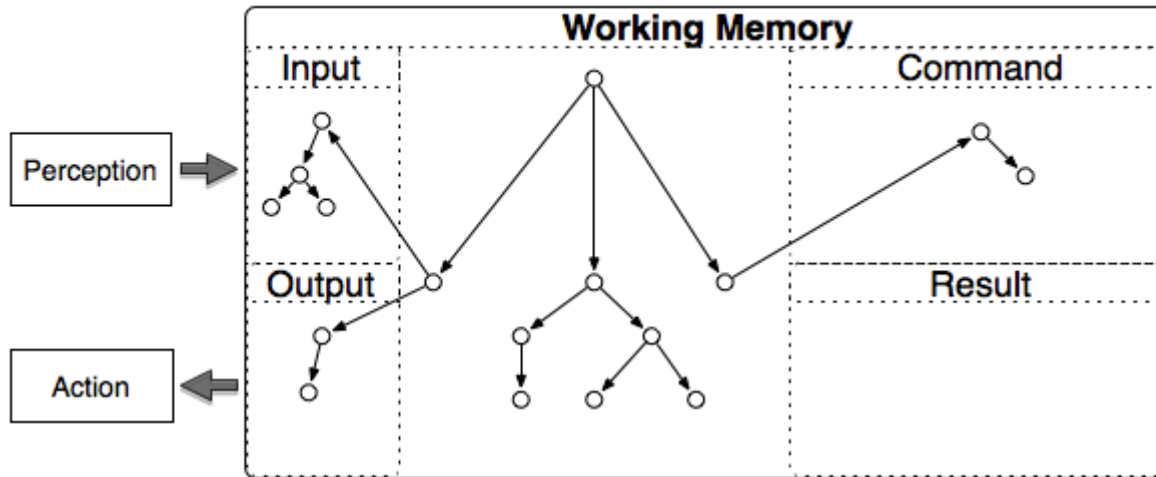


Architectural Integration



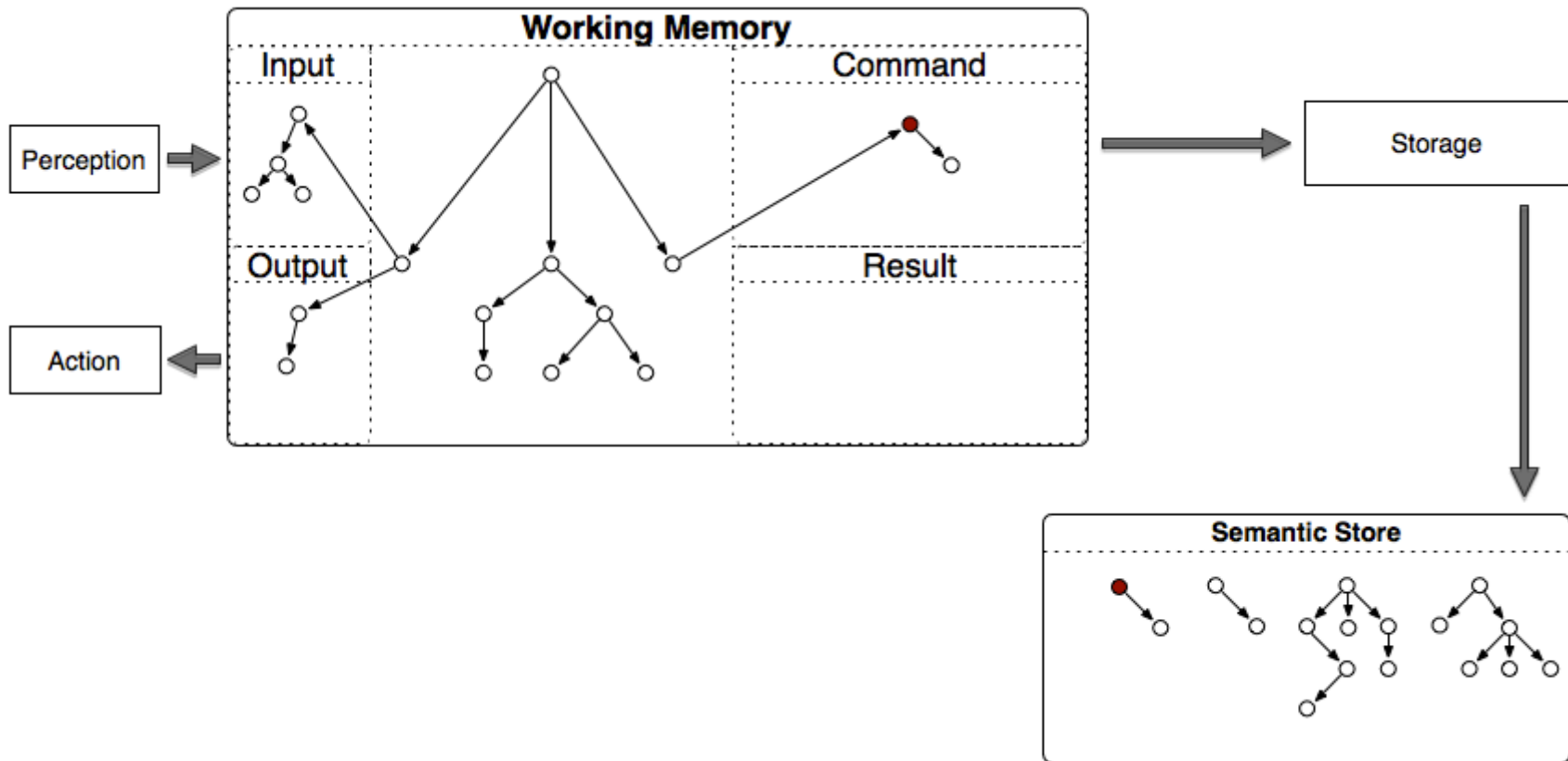
Architectural Integration

Storage



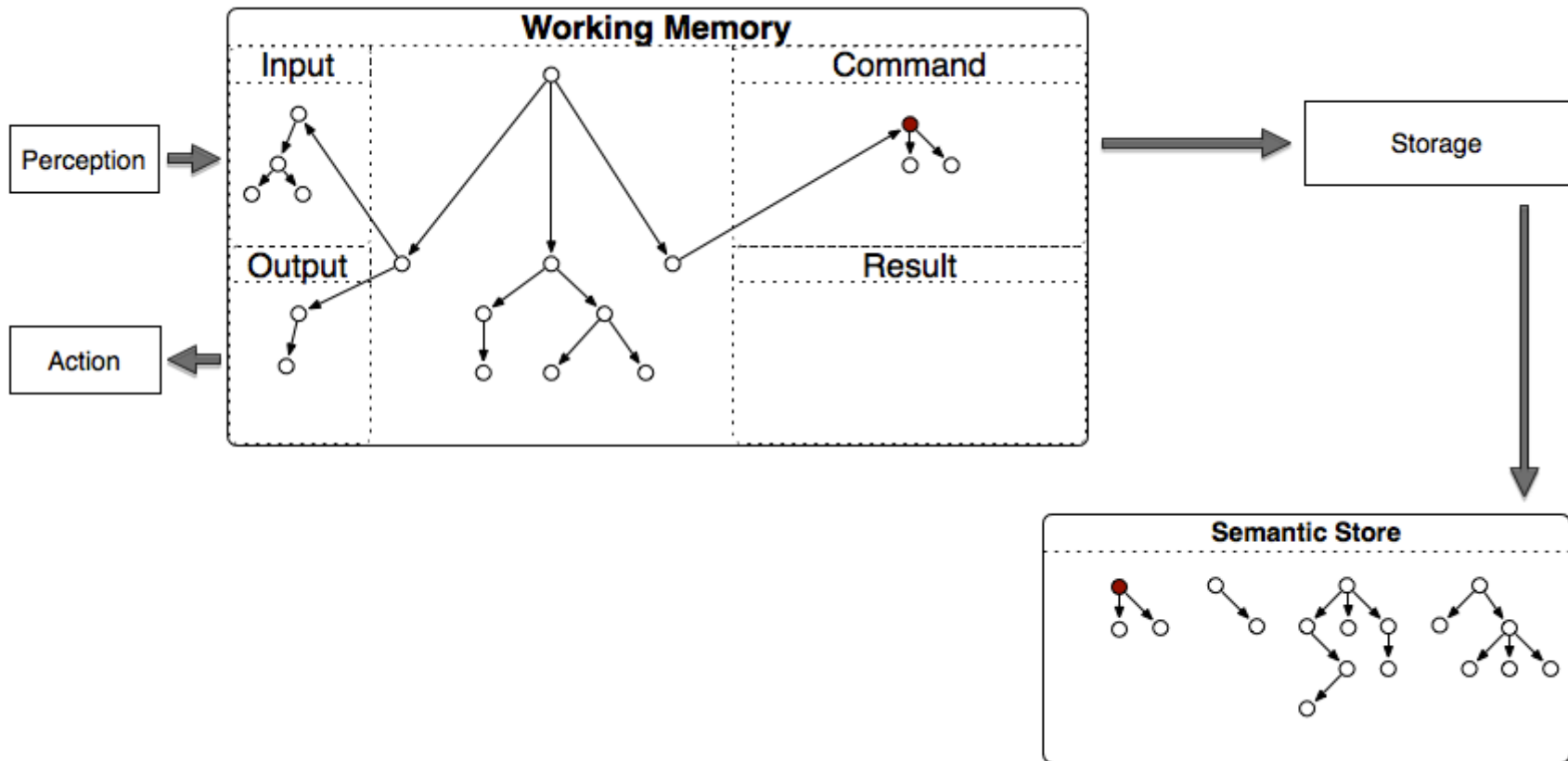
Architectural Integration

Storage



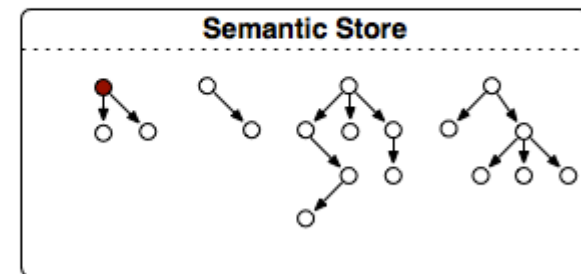
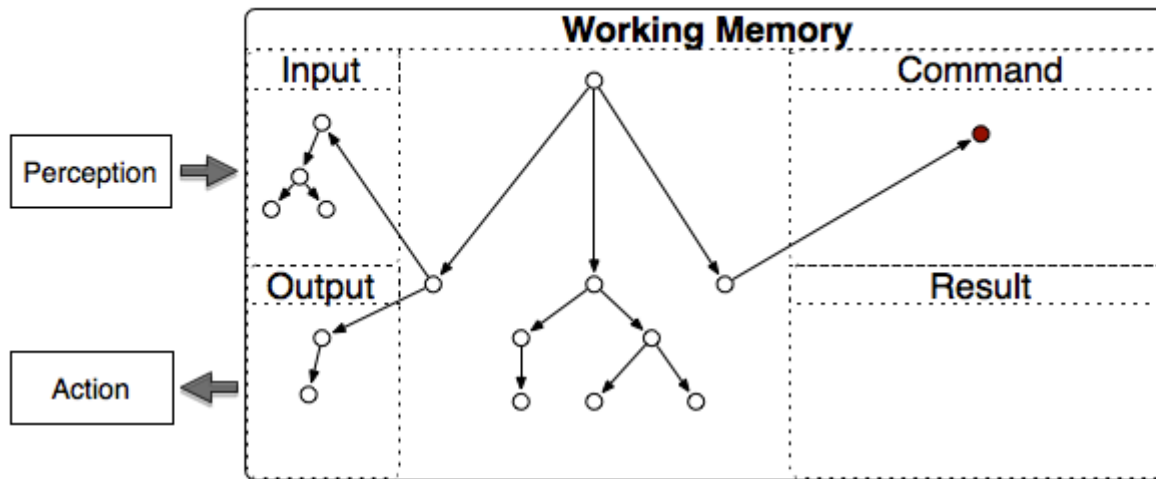
Architectural Integration

Storage



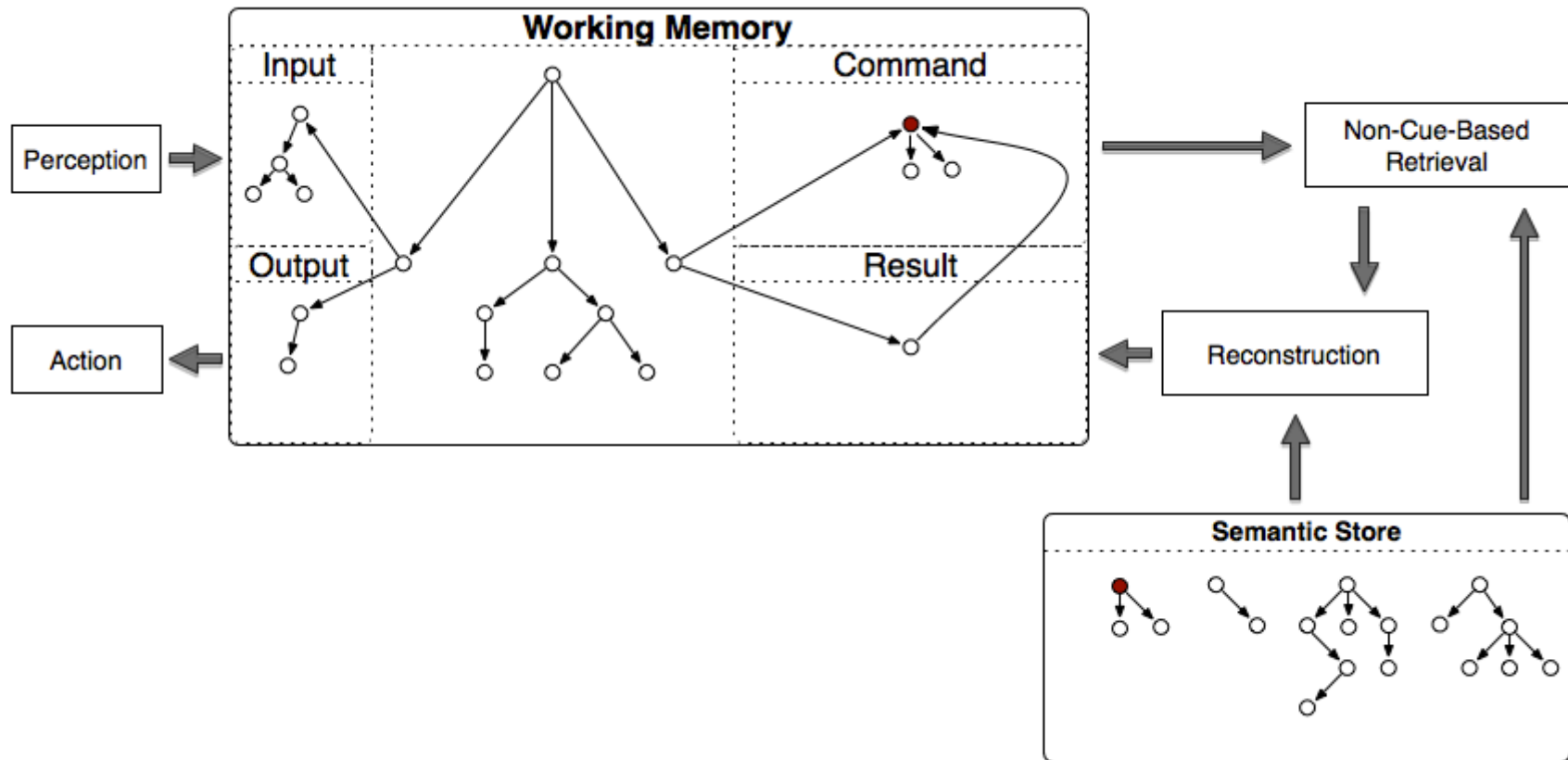
Architectural Integration

Non-Cue-Based Retrieval



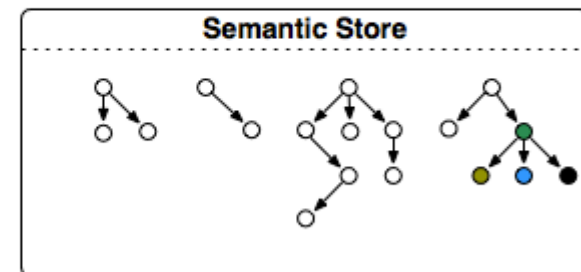
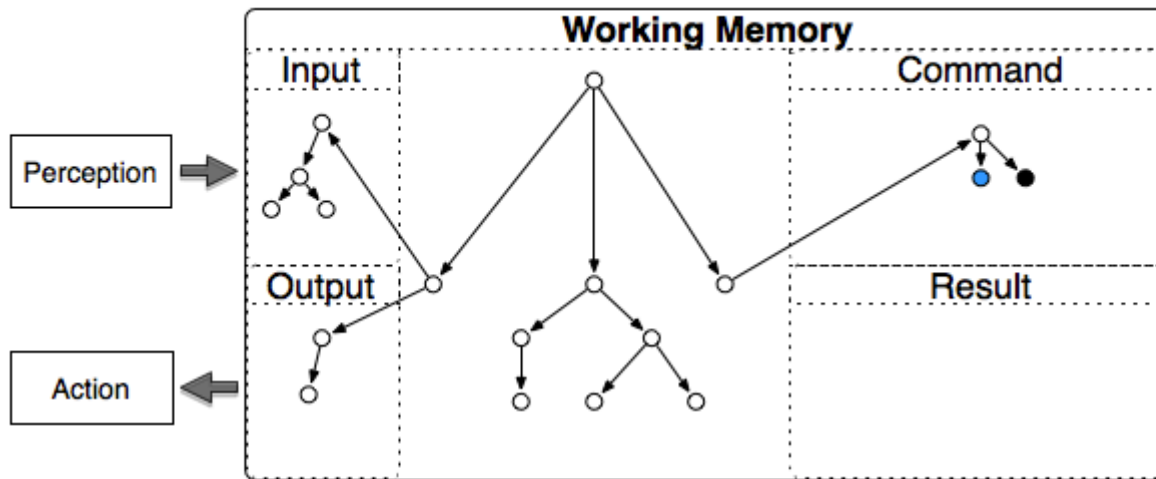
Architectural Integration

Non-Cue-Based Retrieval



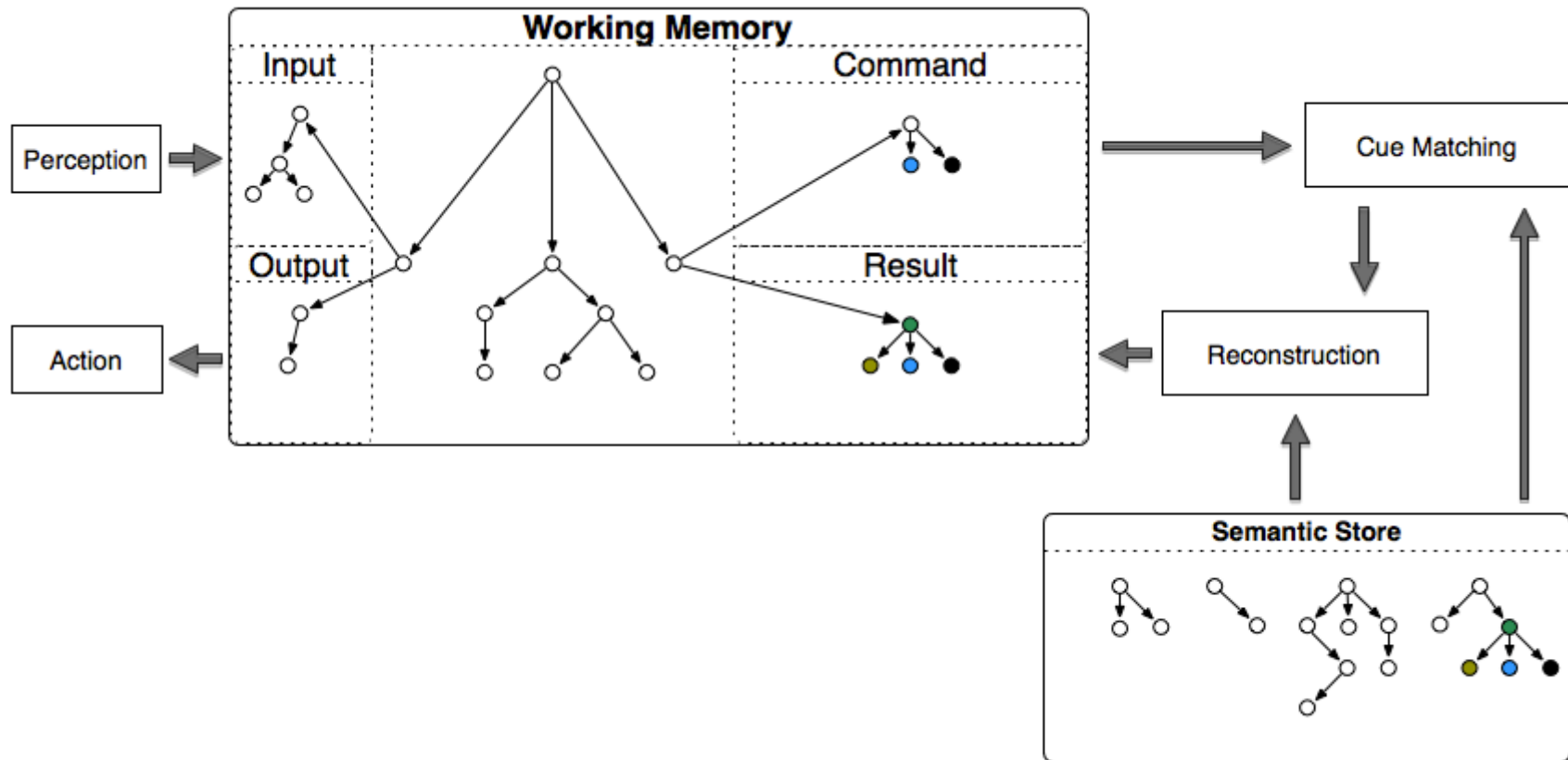
Architectural Integration

Cue-Based Retrieval



Architectural Integration

Cue-Based Retrieval



Basic Usage

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

Working Memory Structure

Soar creates an smem structure on each state

– Soar Java Debugger

- `step 5`
- `print --exact (* ^smem *)`
- `print s2`

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 (ex. S5 or C7) is permanently associated with the identifier
 - When printed, long-term identifiers are prefaced with the @ symbol (ex. @S5 or @C7)
 - When depicted, long-term identifiers are double circles
- Attributes cannot be identifiers (currently)
- The resulting graph is not necessarily connected

Controlling Semantic Memory

Get/Set a parameter:

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

SMem is **disabled** by default. Try enabling it...

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

Storing Knowledge

Manual

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

Agent

Deliberate (i.e. 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>)  
    ...  
}
```

Manual Storage: Example

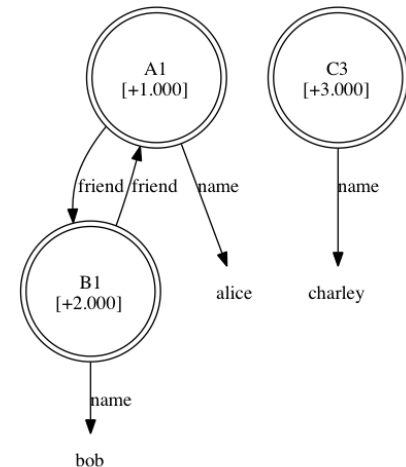
friends-manual.soar

Soar Java Debugger

```
smem --add {  
  (<a> ^name alice ^friend <b>)  
  (<b> ^name bob ^friend <a>)  
  (<c> ^name charley)  
}
```

2. smem --print

3. ctf temp.gv smem --viz



Agent Storage

Syntax

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

- Requires that SMem is enabled (slide 16)
- Processed at end of phase in which rule fires
- Multiple identifiers may be stored at once
- Storage is **not** recursive

Result

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

Agent Storage: Example

friends-agent.soar

- Soar Java Debugger

1. smem --set learning on
2. watch 5
3. →
4. run 4 -p
5. print --depth 10 s2
6. smem --print

```
sp {propose*init
    (state <s> ^superstate nil
        -^name)
-->
    (<s> ^operator <op> +)
    (<op> ^name init)
}
```

```
sp {apply*init
    (state <s> ^operator.name init
        ^smem.command <cmd>)
-->
    (<s> ^name friends)
    (<cmd> ^store <a> <b> <c>)
    (<a> ^name alice ^friend <b>)
    (<b> ^name bob ^friend <a>)
    (<c> ^name charley)
}
```

Examining the Trace

```
=>WM: (25: C3 ^name charley)
=>WM: (24: B1 ^friend A1)
=>WM: (23: B1 ^name bob)
=>WM: (22: A1 ^friend B1)
=>WM: (21: A1 ^name alice)
=>WM: (20: C2 ^store A1)
=>WM: (19: C2 ^store B1)
=>WM: (18: C2 ^store C3)
=>WM: (17: S1 ^name friends)
--- Change Working Memory (PE) ---
=>WM: (28: R3 ^success @A1)
=>WM: (27: R3 ^success @B1)
=>WM: (26: R3 ^success @C3)
```

Agent Storage: Modification

friends-agent-mod.soar

Rules

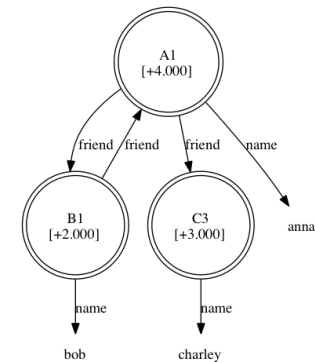
```
sp {propose*mod
  (state <s> ^name friends)
-->
  (<s> ^operator <op> +)
  (<op> ^name mod)
}

sp {apply*mod
  (state <s> ^operator.name mod
    ^smem.command <cmd>)
  (<cmd> ^store <a> <b> <c>)
  (<a> ^name alice)
  (<b> ^name bob)
  (<c> ^name charley)
-->
  (<a> ^name alice -)
  (<a> ^name anna
    ^friend <c>)
  (<cmd> ^store <b> -)
  (<cmd> ^store <c> -)
}
```

Result

1. smem --set learning on
2. run -p 4
3. run -p 5
4. smem --print

```
(@A1 ^friend @B1 @C3 ^name anna [+4.000])
(@B1 ^friend @A1 ^name bob [+2.000])
(@C3 ^name charley [+3.000])
```



Semantic Store Statistics

- Soar Java Debugger
 1. Source `friends-manual.soar`
 2. `smem --stats`
 - Nodes: number of long-term identifiers
 - Edges: number of features/relations
 - Stores: number of agent stores

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:

- Requires that SMem is enabled (slide 16)
- Only one per state per decision
- Processed during *output* phase
- Only re-processed if WM changes to commands
 - Meta-data (status, etc) automatically cleaned by the architecture

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)

Non-Cue-Based Retrieval: Example

ncb-retrieval.soar

- Soar Java Debugger
 1. `smem --set learning on`
 2. `smem --add {`
 `(@A1 ^name alice ^friend @B1 @C3)`
 `(@B1 ^name bob ^friend @A1)`
 `(@C3 ^name charley)}`
 3. `sp {ncb`
 `(state <s> ^superstate nil`
 `^smem.command <cmd>)`

 `-->`
 `(<cmd> ^retrieve @A1)}`
 4. `run 5 -p`
 5. `print --depth 10 s2`
 6. `smem --stats`

Non-Cue-Based Retrieval: Debrief

- Be cautious of long-term identifiers in rules
 - Only legal if already in semantic store
 - Will occur via chunking
- Only features/relations of @A1 added to WM
 - Features/relations of @B1, @C3 would require additional `retrieve` commands
- Statistics kept about number of `retrieve` commands processed
 - `smem --stats`
 - (“Retrieves”)
- Meta-data maintained during *output* phase
 1. `excise ncb`
 2. `run 2 -p`
 3. `print --depth 10 s2`
 4. `run 3 -p`
 5. `print --depth 10 s2`

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>)  
(<result> ^<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)

- See `activation-mode` parameter in Manual
- When you execute `smem -p/v`, the bias value is indicated

Non-Cue-Based Retrieval: Example

cb-retrieval.soar

- Soar Java Debugger

1. `smem --set learning on`
2. `smem --add {`
 `(@A1 ^name alice ^friend @B1 @C3)`
 `(@B1 ^name bob ^friend @A1)`
 `(@C3 ^name charley)}`
3. `sp {cbr`
 `(state <s> ^superstate nil`
 `^smem.command <cmd>)`

 `-->`
 `(<cmd> ^query.name alice)}`
4. `run 5 -p`
5. `print --depth 10 s2`
6. `smem --stats`

Prohibition

Cue-based retrievals can optionally prohibit the retrieval of one-or-more long-term identifiers

Syntax

(<smem> ^command <cmd>)

(<cmd> ^prohibit <l*ti*-1> <l*ti*-2> ...)

Prohibition: Example

prohibit.soar

- Soar Java Debugger

1. `smem --set learning on`
2. `smem --add {
 (@A1 ^name alice ^friend @B1 @C3)
 (@B1 ^name bob ^friend @A1)
 (@C3 ^name charley)}`
3. `sp {prohibit
 (state <s> ^superstate nil
 ^smem.command <cmd>)
-->
 (<cmd> ^query.name <some-name>
 ^prohibit @A1 @C3)}`
4. `run 5 -p`
5. `print --depth 10 s2`

WordNet Demo

<http://code.google.com/p/soar/wiki/WordNet>

- Scripts to convert WN-LEXICAL to SMem
 - Output: `smem --add { ...`
 - >821K long-term identifiers, >3.97M edges, ~88MB
 - Source: ~5-10 minutes, ~1GB memory
- SMem uses a SQLite backend
 - Has the ability to save semantic stores to disk and use disk-based databases
 - `smem --backup filename`
 - `smem --set path filename`

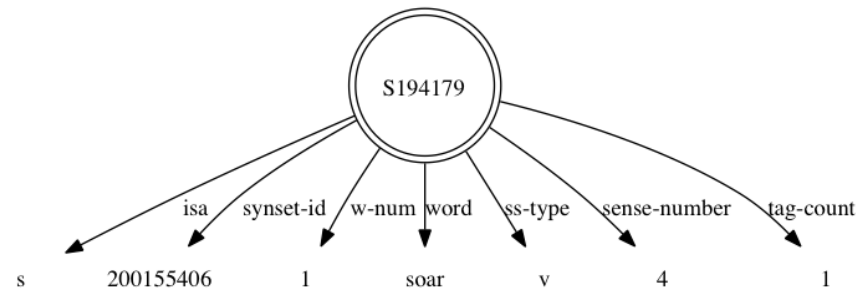
WordNet: Make Disk Store

- Soar Java Debugger
 - `source wn.soar`
 - ~5-10 minutes
 - `smem --stats`
 - `smem --backup path/to/filename.db`
 - ~1 minute
- Soar Java Debugger
 - `smem --set path path/to/filename.db`
 - `run 1 -e`
 - ~0.5 seconds
 - `smem --stats`

WordNet: Representation

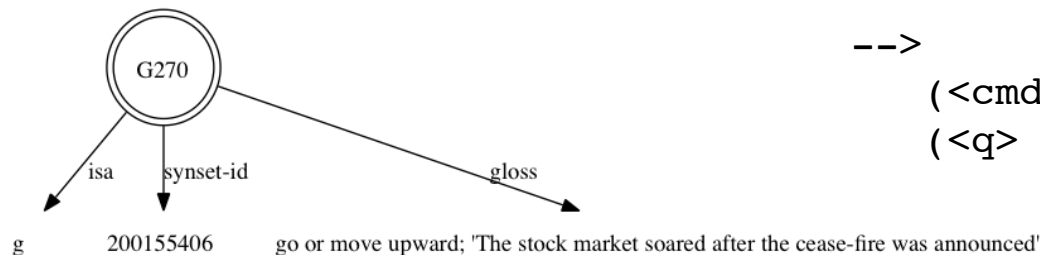
“sense” of the “verb” to “soar”

```
sp {soar*v
  (state <s> ^superstate nil
    ^smem.command <cmd>)
-->
(<cmd> ^query <q>)
(<q> ^ss-type |v|
  ^word |soar|
  ^isa s)
```



“gloss” with the “synset-id” 200155406

```
sp {soar*v*gloss
  (state <s> ^superstate nil
    ^smem.command <cmd>)
-->
(<cmd> ^query <q>)
(<q> ^isa g
  ^synset-id 200155406)
```



WordNet: Task

wn-senses.soar

Find all definitions, given lexical word/POS

- Use `wn-senses-start.soar` as a baseline
- High-level algorithm

1. query: `^isa s ^word lex ^ss-type pos`
2. If successful
 - a) query: `^isa g ^synset-id <sense ^synset-id>`
 - b) If successful
 - » write `<gloss ^gloss>`
 - c) prohibit: `<sense>`
 - d) Loop
3. Else
 - a) (halt)

Additional Resources

- Documentation
- Demo agents
- Readings

Documentation

Manual

`share/soar/Documentation`

Additional Topics

- Details of integration with other mechanisms
- Retrieval biases
- Performance
- Usage: commands, parameters, statistics, etc.
- ...

Demo Agents

share/soar/Demos

- Arithmetic

Performs rule-based addition/subtraction using either working memory or semantic memory as a store of facts

Select Readings

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

2006

- Integrating Semantic Memory into a Cognitive Architecture
 - Yongjia Wang, John E. Laird (Technical Report)

2010

- Extending Soar with Dissociated Symbolic Memories
 - Nate Derbinsky, John E. Laird (AISB)
- Towards Efficiently Supporting Large Symbolic Memories
 - Nate Derbinsky, John E. Laird (ICCM)