

# New Debugging Tools in TSI 3.0

Mazin Assanie

University of Michigan Artificial  
Intelligent Lab



# What's new in TSI 3.0

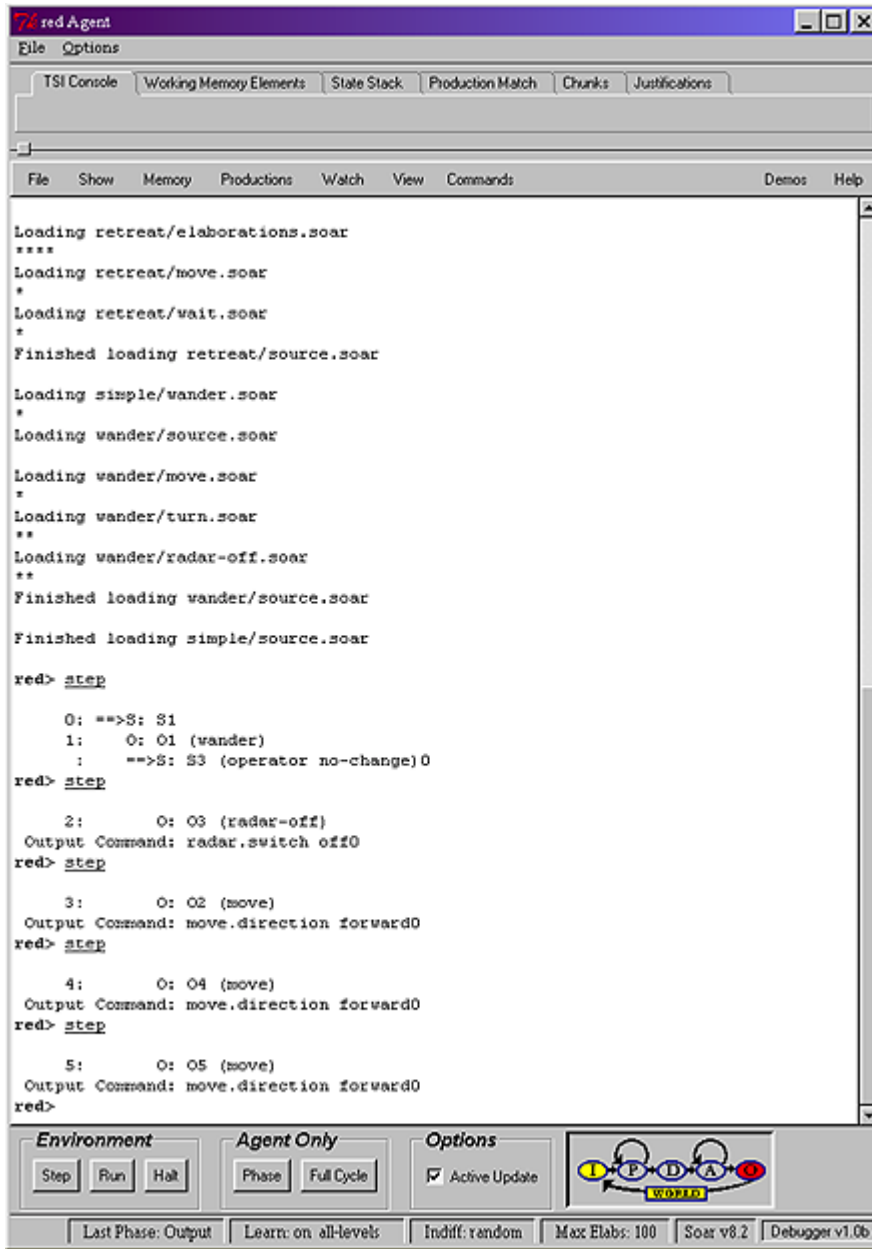
- Look at Soar working memory more intuitively: Tree views of working memory, state map, production matches
- Graphical representation of decision cycle
- Traditional TSI interface maintained
- Generalized interface to Tcl environments now part of the TSI
- Bug fixes



# Compatibility

- New graphical views require Tcl 8.0 or newer.
- Backward compatible
  - Old TSI interface is a component of TSI 3.0
  - Old TSI can be used exclusively by setting global variable `tsiConfig(interfaceType)` to 1



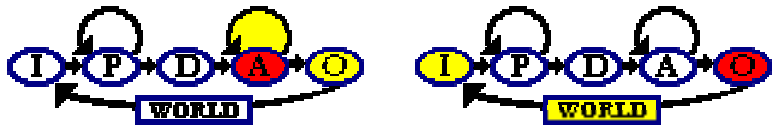
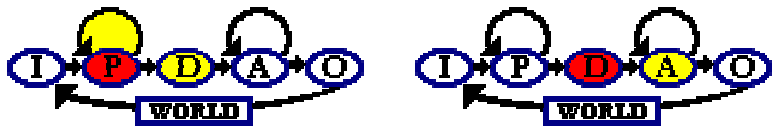
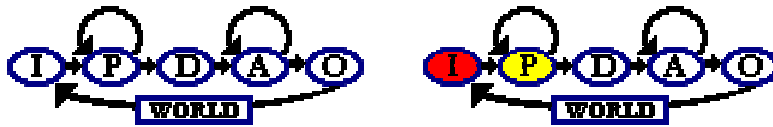


# General Description

- Notebooks
- Split pane
- Graphical decision cycle
- Step buttons
- Active update
- Configuration Settings

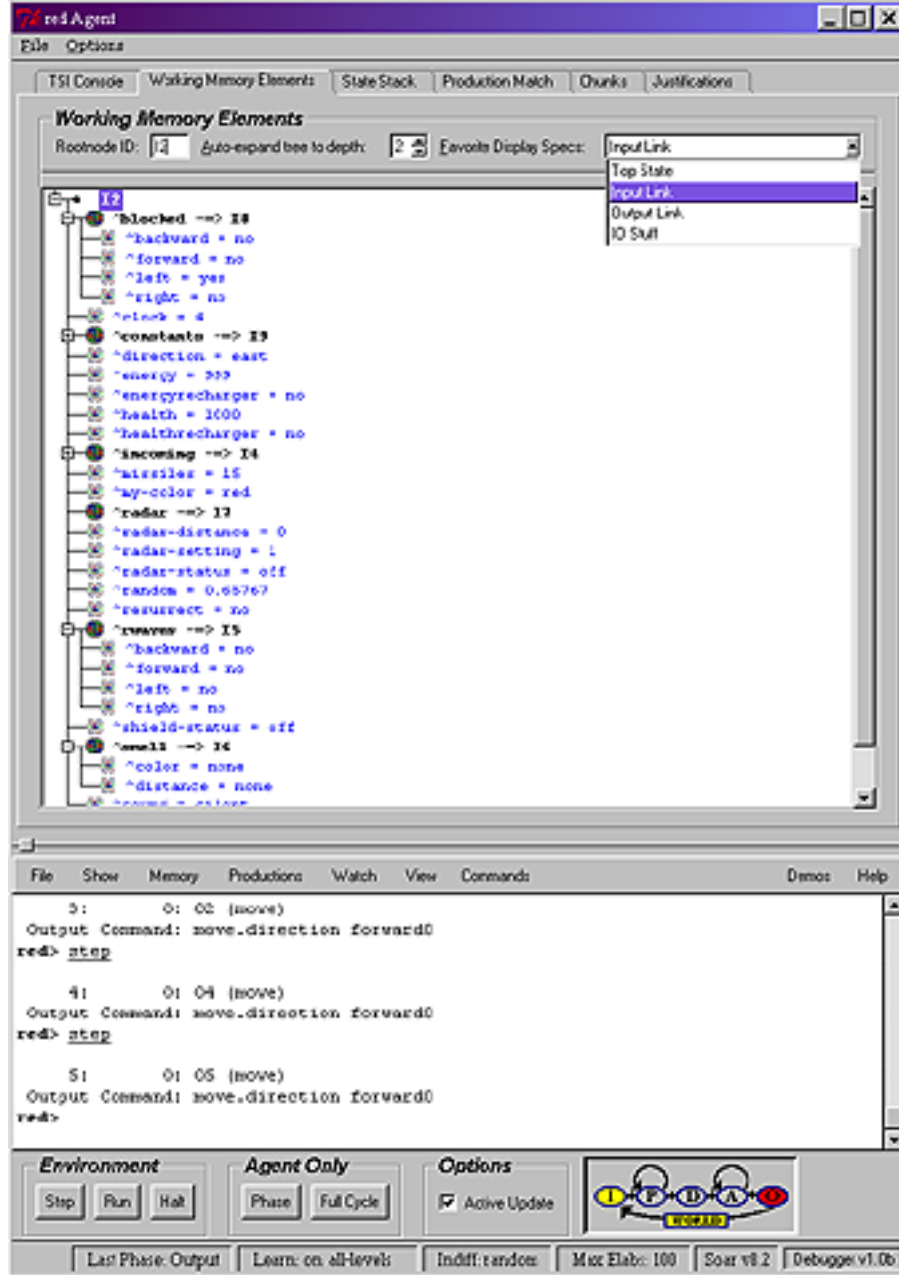


# Graphical Decision Cycle



- Shows
  - Last phase
  - Possible next phases





# The WME Tree View

- Tree starting at any specified WME
- Auto-expansion
- Persistent list of favorite display specifications



The screenshot shows the 'red Agent' software interface. At the top, there are menu options 'File' and 'Options'. Below that, a series of tabs: 'TSI Console', 'Working Memory Elements', 'State Stack', 'Production Match', 'Chunks', and 'Justifications'. The 'State Stack' tab is active, displaying a tree view of states and operators. The root node is 'StateStack', which contains several sub-nodes: '<S1>: O1 wander', 'Operator ==> O1', '<S3>: O5 move', and several attribute nodes like '^attribute = operator', '^choices = none', '^lapasse = no-change', '^io ==> I1', '^name = wander', '^operator ==> O5', '^quiescence = t', '^superstate ==> S1', and '^type = state'. Below the State Stack view is a console window with a menu bar 'File Show Memory Productions Watch View Commands Demos Help'. The console displays the following text:

```

Loading retreat/elaborations.soar
****
Loading retreat/move.soar
*
Loading retreat/wait.soar
*
Finished loading retreat/source.soar

Loading simple/wander.soar
*
Loading wander/source.soar

Loading wander/move.soar
*
Loading wander/turn.soar
**
Loading wander/radar-off.soar
**
Finished loading wander/source.soar

Finished loading simple/source.soar

red> step

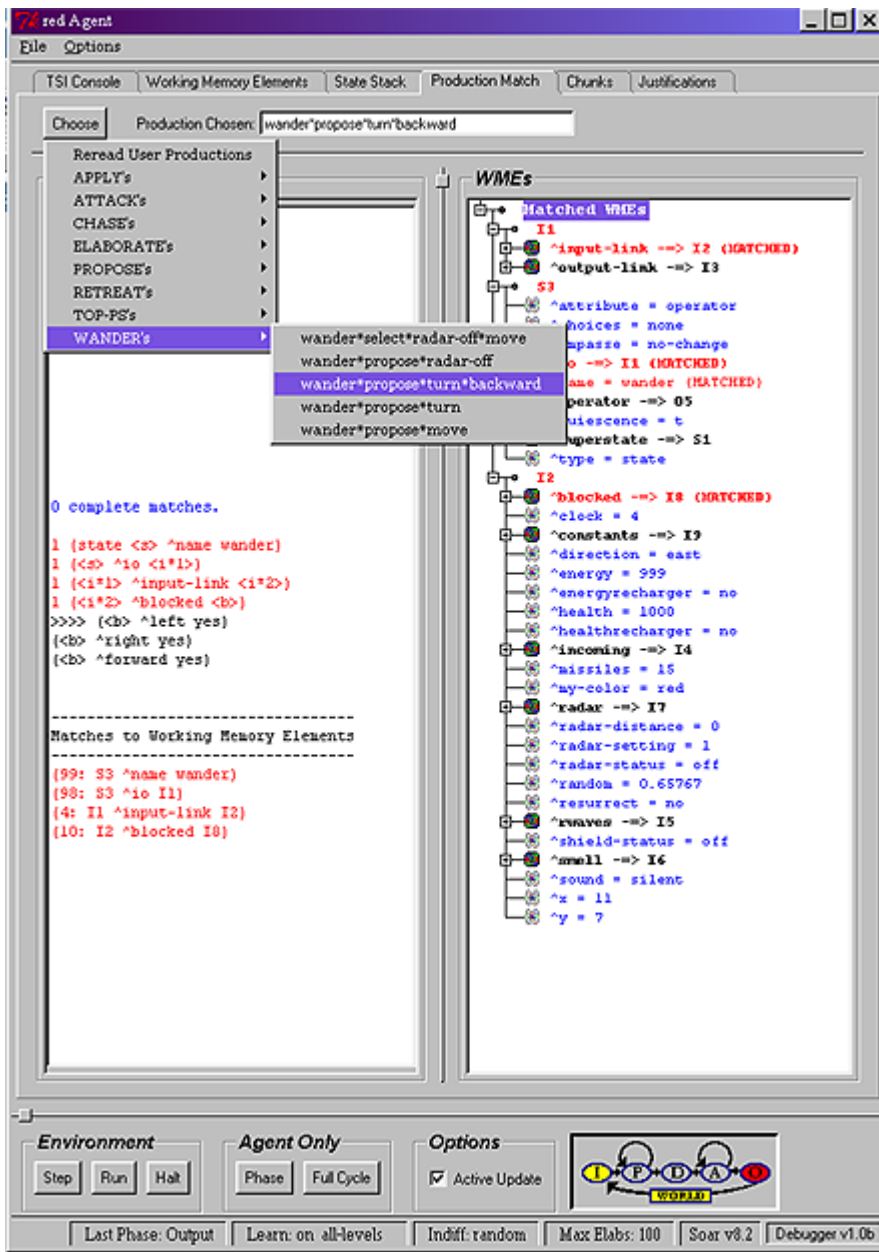
0: ==>S: S1
1: 0: O1 (wander)
   : ==>S: S3 (operator no-change)0
  
```

At the bottom of the interface, there are control panels for 'Environment' (Step, Run, Halt), 'Agent Only' (Phase, Full Cycle), and 'Options' (Active Update checked). A small diagram with nodes 'I', 'P', 'D', 'A', 'G' and a 'WORKER' label is also present. The status bar at the very bottom shows: 'Last Phase: Output | Learn: on all-levels | Indiff: random | Max Elabs: 100 | Soar v8.2 | Debugger v1.0b'.

# The State Stack View

- List of states with currently selected operator
- Works like standard WME tree





# The Production Match View

- Select production from hierarchical list of user productions
- Matches output in left pane
- Matched WMEs in right pane





# Chunks and Justifications View

The screenshot displays the 'red Agent' software interface. The main window is titled 'All Chunks This Agent Has Compiled' and contains three chunks of code:

```
sp (chunk-1*d2*opnochange*1
:chunk
(state <s1> ^operator <o1> ^io <i1>)
<o1> ^name wander)
<i1> ^input-link <i2> ^output-link <o2>)
<i2> ^radar-status on ^radar <r1>)
<r1> ^^( <a1> << energy health missiles tank >> ) <a2>)
-->
<o2> ^radar <r2> +)
<r2> ^switch off +)
)

sp (chunk-2*d3*opnochange*1
:chunk
(state <s1> ^operator <o1> ^io <i1>)
<o1> ^name wander)
<i1> ^output-link <o2> ^input-link <i2>)
<i2> ^blocked <b1>)
<b1> ^forward no)
-->
<o2> ^move <m1> +)
<m1> ^direction forward +)
)

sp (chunk-3*d3*opnochange*2
:chunk
(state <s1> ^operator <o1> ^io <i1>)
<o1> ^name wander)
<i1> ^output-link <o2> ^input-link <i2>)
<o2> ^radar <r1>)
<r1> ^status complete)
<i2> ^blocked <b1>)
<b1> ^forward no)
-->
<o2> ^radar <r1> -)
)
```

Below the code is a 'Output Command' window showing a sequence of commands and their outputs:

```
Output Command: move.direction forward
red> stop
0
red>
11:      0: 011 (move)
Output Command: move.direction forward
Output Command: move.direction forward
red>
red> |
```

The bottom of the interface features a control panel with buttons for 'Step', 'Run', 'Halt', 'Phase', and 'Full Cycle'. There is also a checkbox for 'Active Update' and a small diagram of a cycle. The status bar at the very bottom shows 'Last Phase: Output', 'Learn: on all-levels', 'Indiff: random', 'Max: Elabs: 100', 'Soar v9.2', and 'Debugger v1.0b'.



# Nuggets

- Graphical decision cycle useful for both users just learning about the Soar decision cycle and debugging tricky phase-related bugs
- Less wear and tear on your ‘p’ key



# Coals

- Performance hit
  - Depends on amount of information displayed
- Requires more screen real estate
  - Suggest saving your own window prefs
  - Will try to reduce in a future version
  - Possible weird sash positioning
- Need to use Tcl 8
- Be careful with preferences
  - Novice users can mess up



# Where to get

- Soar home page at the University of Michigan
- <http://ai.eecs.umich.edu/soar/projects.html>
- Part of the latest release of Soar, Tanksoar and Eaters

