Proceedings of EuroSoar-10

Technische Universität Berlin

Preface

EuroSoar-10 was held on November, 16, 1996 at the Berlin Insitute of Technology (Technische Universität Berlin). A tutorial (Soar7) was organized and given by R. Young and F. Ritter. EuroSoar was organized as part of the first European Workshop on Cognitive Modeling (Nov. 14-16, 1996). Some papers (e.g., from F. Ritter, T. Johnson, J. Krems) went into this workshop. This is one reason why the number of presentations and the whole structure of EuroSoar considerably differed from previous meetings.

Josef F. Krems, Chemnitz.

Content

U. of Nottingham Update - Ritter, F.E.; Gordon, C.B.; Jones G. 03
Soar Modelling and Rational Modelling of Exploratory Choice – Young, R.M. 04
Strategy Data in the Kanfer-Ackermann Air Traffic Controller Task – John, B.E. 09
Soaring back to lisp – Aasman, J. 32
U. of Nottingham Update

Frank E. Ritter, Gordon D. Baxter, and Gary Jones
with some work with Randy Jones and Tony Kalus
16 Nov 1996

- Mirrors of CMU and ISI sites up and running
  (all available through http://www.psysc.nott.ac.uk/users/ritter.html)

- Soar Frequently Asked Questions (FAQ) released quarterly,
  next release will include pointers to more teaching materials

- Able, III available as an example model and tool

- Psychological Soar Tutorial updated to 7.0.4

- Tcl/Tk Soar Interface available with Able, III

- Ship radar task simulator set up

- Eye/hand specified and being built to tie Soar to simulation.

- Set up to compare SCA variants with 80 subjects' data to study
  acquisition of categorisation (can read files of stimuli in and write out
  trial output)
Soar Modelling and Rational Modelling of Exploratory Choice

Richard M Young

MRC Applied Psychology Unit
Cambridge, England

Talk presented at 10th EuroSoar Workshop
(in conjunction with Workshop on Cognitive Modelling)
Berlin
16th November 1996
Reminder: IDXL

Last year, at both US Soar and EuroSoar workshops, I talked about the IDXL model (Rieman, Young & Howes, 1996, *International Journal of Human-Computer Studies*, 44, 743-775) of exploratory learning on the Cricket Graph task.

- The task had experienced Macintosh users, but who had never used a graphing program before, use Cricket Graph to draw a graph in a specified style from a data file.

- IDXL is built around incremental comprehension. It scans the display items, performing quanta of comprehension. Information about an item cumulates in the form of comprehension chunks.

- IDXL exhibits key empirical phenomena, such as
  - repeated scanning of items
  - progressive focussing on a subset of items
  - iterative deepening of attention.

Quasi-Rationality of IDXL

Notice that the iteratively deepening attention is, at least in an informal sense, *rational*:

- doesn't make sense to commit much effort early to a route that may not be right
- searching more broadly may reveal an obviously good route elsewhere

- We had always thought about IDXL in terms of rational analysis. The intention was for the model to prove a discrete, symbolic approximation to a "rational" optimisation of costs and benefits:
  - by using incremental comprehension, with internal and external steps tried in order of increasing cost;
  - by insisting on more comprehension before committing to action, the more costly or risky the action.

- So the question arises: Does IDXL indeed provide a discrete, symbolic approximation to the behaviour of the rationally optimal strategy?

- In order to answer that question, I had to perform a rational analysis of the task.

- To anticipate the conclusion: We think, yes, it could, but not IDXL as we originally constructed it.
Statement of Problem for Rational Analysis

- The subject has to choose one of N possible moves $M_1, M_2, \ldots, M_N$.
- For each move $M_i$, there is available a set of assessment procedures $A_{ij}$, each with a cost $B_{ij}$.
  - The $k$th assessment chosen is written $A_k$ with cost $B_k$.
- When $A_{ij}$ applied to $M_i$, it yields an assessment of $M_i$'s relevance to the goal, $R_i$.
- The set of $R_i$ determine (through a mapping to be discussed) the set of probability estimates $P_i$. One move is correct: $\sum P_i = 1$.
- Associated with move $M_i$ is a payoff $Y_i = P_i \cdot (G+C)$.
- After a series of $n$ assessments $A_1, A_2, \ldots, A_n$, the subject chooses the move $M_m$ with maximal payoff $Y_m$. The worth of that move is $W = Y_m - \sum_{k=1}^{n} B_k$.
  - i.e. $W = P_m \cdot (G+C) - \sum B_k$.
- Primary task for rational analysis is to identify a strategy for choosing an optimal sequence of assessments $A_k$ and a stopping point $n$, to maximise the expected value of $W$.

Optimum Strategy

- We have for the worth of the $n$th assessment:
  $W(n) = Y_m(n) - \sum_{k=1}^{n} B_k$. If we consider doing some further assessment $A_a$, the expected improvement in the best worth is
  $\text{Est}(W(a)) - W(n) = [\text{Est}(\max(Y_i(a)) - \max(Y_i(n))] - B_a$
  $= \text{Est}(A_a \max(Y_i)) - B_a$
- If $A_a$ is the next in a series of optimal choices of assessment, what can we say about its choice? In analogous situations in probabilistic search, and in particular for the case of problem solving [proof by John Anderson, personal communication], the optimal choice is the most efficient step, i.e. the one which gives the greatest increase in payoff for the least cost.

The optimal strategy is to choose at each stage the assessment $A_a$ with the greatest efficiency, as measured by the ratio $E(A_a \max(Y_i)/B_a$.

- Stop when the efficiency falls below 1.
### Results of Simulation

#### Lexical, Semantic, Pulldown, Anticipation

### Fit of the Model to Data

<table>
<thead>
<tr>
<th>Empirical data</th>
<th>IDXL</th>
<th>Rat. anal.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses labels that match task description</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Exact matches are fastest, synonyms are slower</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Inferences take still longer</td>
<td>Model doesn’t use inference on labels, but could be another comprehension step</td>
<td>✓</td>
</tr>
<tr>
<td>Meaningless labels require instruction</td>
<td>Takes instruction, but not for labels</td>
<td>not modelled</td>
</tr>
<tr>
<td>Poor labels interact with large number of objects to produce longer search times</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Menus and dialogue boxes easier than direct manipulation</td>
<td>Difference not modelled</td>
<td>not modelled</td>
</tr>
<tr>
<td>Repeated scans with iteratively deepening attention for poor labels</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Menu bar items are tried more freely than pulldown menu items</td>
<td>✓</td>
<td>not modelled</td>
</tr>
<tr>
<td>Initial scanning is guided predominantly by menu layout (left-to-right, etc.)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Obviously inappropriate labels are avoided</td>
<td>Mouse is moved only to potential action candidates</td>
<td>✓</td>
</tr>
<tr>
<td>Individuals vary in exploratory freedom</td>
<td>Adjustable by preferences</td>
<td>Sensitive to numerical assumptions</td>
</tr>
<tr>
<td>May choose without examining all items</td>
<td>Not unless lose other properties</td>
<td>✓</td>
</tr>
</tbody>
</table>
Rational Analysis Throws Light on IDXL

- Rational analysis mostly concurs with IDXL (and with the empirical data)

- Questions whether IDXL is right to "pursue attractive options"
  - analysis shows that expected improvement from further assessing a clear winner is low
  - however, the full story is complicated and IDXL is not obviously wrong

- Suggests that IDXL gone too far in extreme locality of processing. IDXL treats the options independently, but analysis shows that can't get correct rational behaviour that way.

- On the other hand, only three global quantities are involved in the computation
  - value of best relevance, $R_v$
  - value of second best relevance, $R_v$
  - total relevance $\Sigma R$, or something equivalent to it such as the mean relevance $\text{avg}(R)$

- Suggests modifications to IDXL to make it a closer symbolic, qualitative approximation to rational optimum.

Conclusions

- Rational modelling complements the Soar modelling

- Rational analysis shows to what extent IDXL is, and is not, a good symbolic, qualitative approximation to rational strategy.

- Rational analysis
  - illuminates the workings of IDXL
  - provides a theoretical basis, which is currently missing or weak
  - suggests possible improvements to IDXL.
Strategy Data in the Kanfer-Ackerman Air Traffic Controller Task©

Bonnie E. John
Departments of Computer Science and Psychology
and the Human-Computer Interaction Institute
Carnegie Mellon University

Work done with Yannick Lallement

This work is supported by the Office of Naval Research Cognitive Science Program, Contract No. N00014-96-1-0252 96PR01919. Views and conclusions contained in this document are those of the author and should not be interpreted as representing official policies, either expressed or implied, of ONR or the United States Government.
ATC Task:
Land Planes
Following
Many Rules

<table>
<thead>
<tr>
<th>FLT#</th>
<th>TYPE</th>
<th>SCHEDULED</th>
<th>FUEL</th>
<th>POS.</th>
<th>TIME</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>431</td>
<td>DC10</td>
<td>1:35</td>
<td>6</td>
<td>3 n</td>
<td>1:00</td>
<td>1000</td>
</tr>
<tr>
<td>268</td>
<td>747</td>
<td>1:45</td>
<td>6</td>
<td>3 s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>555</td>
<td>prop</td>
<td>1:50</td>
<td>6</td>
<td>3 e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>747</td>
<td>2:00</td>
<td>6</td>
<td>3 w</td>
<td></td>
<td></td>
</tr>
<tr>
<td>912</td>
<td>727</td>
<td>1:15</td>
<td>6</td>
<td>2 n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>prop</td>
<td>1:05</td>
<td>6</td>
<td>2 s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>727</td>
<td>1:20</td>
<td>6</td>
<td>2 e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>157</td>
<td>DC10</td>
<td>1:20</td>
<td>6</td>
<td>1 n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>872</td>
<td>727</td>
<td>1:10</td>
<td>6</td>
<td>1 e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>222</td>
<td>prop</td>
<td>1:00</td>
<td>5</td>
<td>1 w</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Plts in Queue:
<F1> to accept

- Always land into the wind
- Cannot have more than one plane in a single holding level
- Cannot have more than one plane on a single runway
- Can only move one holding level at a time
- 747s must always use a long runway
- Propeller planes can always land on a short runway
- DC10s cannot land on a short runway in wet or icy weather
- 727s cannot land on a short runway in icy weather
- Lose points for every plane landed with less than 3 minutes of fuel
- Lose points for every violation of the rules
- Lose a lot of points for every plane that runs out of fuel
Other Architectures Currently Modeling this Task with ONR Support

- **ACT-R**
  Carnegie Mellon University
  Departments of Psychology & Computer Science
  Lynne Reder & John Anderson - PIs
  Frank Lee - grad student, Chris Schunn - postdoc

- **RATLE**
  Inserting advice into a knowledge-based neural network
  Computer Sciences Department
  University of Wisconsin - Madison
  Jude Shavlik - PI

- **Hybrid of explanation-based and reinforcement learning**
  Computer Science Department
  Oregon State University
  Tom Dietterich - PI
Changes in Strategy for Bringing Planes in from the Queue -- Aggregate Data
(Lee, Anderson & Matessa, 1995)
What are Individuals Doing that Produces that Aggregate Graph?

Looked at timelines of individual queue-acceptance behavior

For example, Subject AIR06253 - trial 1

Each point is a plane brought in from the queue

Seconds since the beginning of the trial (0 to 600)
Observed strategies for bringing plane in from the queue & strategy shifts

- Strategies involving **TIME**
- Strategies involving **LOCATION**
- Strategy **SHIFTS**
  - none
  - gradual
  - abrupt
  - exploring
Strategy Involving TIME: Stacked

For example, Subject AIR06255 - trial 15
Strategy Involving TIME: Opportunistic

For example, Subject AIR06269 - trial 15
Strategy Involving TIME: Sequential

For example, Subject AIR06209 - trial 8
Strategy Involving LOCATION: All 3 Hold-Levels

For example, Subject AIR06262 - trial 4
Strategy Involving LOCATION: 2 Hold-Levels

Subject AIR06287 - trial 17

Subject AIR6214 - trial 9
Strategy Involving LOCATION:
1 Hold-Level

Subject AIR6316 - trial 10

Subject AIR06285 - trial 11

Subject AIR06289 - trial 4
Strategy SHIFTS:
None

For example, Subject AIR06266
Trial 3

...13 more trials

Trial 17

Just speed up learning
Strategy SHIFTS: Gradual

For example, Subject AIR06283 - trials 2 & 3
Strategy SHIFTS:
Abrupt - Between Trials, no break

For example, Subject AIR06256 - trials 2 & 3
Strategy SHIFTS:
Abrupt - Between Trials, short break

For example, Subject AIR06281 - trials 3 & 4
Strategy SHIFTS: Abrupt - Between Trials, overnight

For example, Subject AIR06256 - trials 9 & 10
Strategy SHIFTS:
Abrupt - Within Trial

For example, Subject AIR06213 - trial 13
Subject: AIR06255

Exploring stacks in the midst of a sequential strategy.
What does it mean to Soar models?

- Architecture must not preclude any of these
- Different knowledge & mechanisms (i.e., problem-solving techniques) must account for the differences
- **Intra-trial abrupt strategy shifts:**
  Must have enough cognitive slack time in the midst of this real-time task to deliberate about the task
- **Inter-trial abrupt strategy shifts:**
  Must learn enough about the environment to deliberate without being in front of the comptuer
- **Exploring:**
  - Probably must be able to plan explorations
  - Must retain info across trials

- Other Suggestions?
Gold & Coal

- Neat data!!!!
  - Only preliminary descriptive data so far of strategy use
  - No Soar model yet
### First 2 Trials

<table>
<thead>
<tr>
<th>Location</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>182</th>
<th>183</th>
<th>283</th>
<th>All</th>
<th>TOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIS</td>
<td>122</td>
<td>23</td>
<td>53</td>
<td>52</td>
<td>32</td>
<td>24</td>
<td>15</td>
<td>321</td>
</tr>
<tr>
<td>STA</td>
<td>20</td>
<td>6</td>
<td>22</td>
<td>41</td>
<td>16</td>
<td>50</td>
<td>130</td>
<td>286</td>
</tr>
<tr>
<td>SEQ</td>
<td>68</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>68</td>
</tr>
<tr>
<td>OPP</td>
<td>16</td>
<td>1</td>
<td>0</td>
<td>29</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>82</td>
</tr>
<tr>
<td>TOT</td>
<td>256</td>
<td>30</td>
<td>75</td>
<td>122</td>
<td>48</td>
<td>76</td>
<td>149</td>
<td>756</td>
</tr>
</tbody>
</table>

### Last 3 Trials

<table>
<thead>
<tr>
<th>Location</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>182</th>
<th>183</th>
<th>283</th>
<th>All</th>
<th>TOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIS</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>7</td>
<td>20</td>
</tr>
<tr>
<td>STA</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>58</td>
<td>0</td>
<td>109</td>
<td>87</td>
<td>256</td>
</tr>
<tr>
<td>SEQ</td>
<td>37</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>37</td>
</tr>
<tr>
<td>OPP</td>
<td>267</td>
<td>34</td>
<td>18</td>
<td>92</td>
<td>2</td>
<td>18</td>
<td>12</td>
<td>383</td>
</tr>
<tr>
<td>TOT</td>
<td>245</td>
<td>36</td>
<td>19</td>
<td>153</td>
<td>3</td>
<td>134</td>
<td>106</td>
<td>696</td>
</tr>
</tbody>
</table>

- Shift to identifiable strategies
- Stacks get longer
- Sequential is in a sensible place (level 1) but # trials using it halves
- Opportunistic quadruples, mostly in the lower levels
- Location 3 & 183 reduce substantially
### STABLE STRATEGIES (by people)

#### FIRST 3 TRIALS

<table>
<thead>
<tr>
<th></th>
<th>UN</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>142</th>
<th>143</th>
<th>213</th>
<th>All</th>
<th>TOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>UN</td>
<td>45</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>NIS</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>STA</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>SEQ</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>OPP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TOT</td>
<td>52</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>63</td>
</tr>
</tbody>
</table>

#### LAST 3 TRIALS

<table>
<thead>
<tr>
<th></th>
<th>UN</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>142</th>
<th>143</th>
<th>213</th>
<th>All</th>
<th>TOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>UN</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>NIS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>STA</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>SEQ</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>OPP</td>
<td>5</td>
<td>16</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>TOT</td>
<td>19</td>
<td>19</td>
<td>1</td>
<td>9</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>58</td>
<td>58</td>
</tr>
</tbody>
</table>

- Strategies & Location stabilize
  - 30% → 85%
- Long stacks became stable
- Seq in 1 same (but we don't know if these are the same people yet)
- Opp is the big favorite by the end.
This talk:

• why back to Lisp?
• how does it compare to ....?
• how does it work?
• what needs to be done?
• anyone interested....?
Why back to lisp?
   it is all very personal

- frustration
  - interfacing to external worlds
  - tweaking Soar
  - `&^$C++**&^$tcl/tk*)(* @ %#

- curiosity
  - prolog compilation for soar?

- fun

- However ....
### Compared to the official Soars

<table>
<thead>
<tr>
<th>Base technology</th>
<th>Prolog Compilation Techniques, Overlap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Soar rules are compiled to Lisp terms</td>
</tr>
<tr>
<td></td>
<td>Simple plist based working memory representation</td>
</tr>
<tr>
<td></td>
<td>Rule selection based on attributes added to WM</td>
</tr>
<tr>
<td>Expressivity of rules</td>
<td>Full Soar expressiveness</td>
</tr>
<tr>
<td>Syntax</td>
<td>98% the same (s-exp for sets, no equality)</td>
</tr>
<tr>
<td>Decision mechanism</td>
<td>Soar 6 preference logic</td>
</tr>
<tr>
<td>Truth Maintenance</td>
<td>Yes</td>
</tr>
<tr>
<td>Input/Output</td>
<td>Yes</td>
</tr>
<tr>
<td>Chunking</td>
<td>Yes, including negated conjunctions and very fast duplicate chunk detection (constant time by 2 step comparison)</td>
</tr>
<tr>
<td>Lisp code size</td>
<td>~93 000 (vs 768 791 in Soar522)</td>
</tr>
<tr>
<td>Dynamic memory size</td>
<td>~1500 bytes per Soar rule</td>
</tr>
<tr>
<td>Size</td>
<td>No additional run-time costs</td>
</tr>
<tr>
<td>Speed - 10 rules</td>
<td>more than 10 times as fast (compares to C code for small systems)</td>
</tr>
<tr>
<td>Speed - large systems</td>
<td>?????, but expected to rise</td>
</tr>
</tbody>
</table>
Comparing sizes

<table>
<thead>
<tr>
<th></th>
<th>Soar 522</th>
<th>My Soar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defstructs</td>
<td>24</td>
<td>7</td>
</tr>
<tr>
<td>Defuns</td>
<td>1611</td>
<td>172</td>
</tr>
<tr>
<td>Defmacros</td>
<td>189</td>
<td>17</td>
</tr>
<tr>
<td>Eval Whens</td>
<td>41</td>
<td>0</td>
</tr>
<tr>
<td>Defvars</td>
<td>282</td>
<td>68</td>
</tr>
<tr>
<td>Defconstants</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>Defparameter</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>Proclaims</td>
<td>832</td>
<td>4</td>
</tr>
<tr>
<td>Defdsmmacro</td>
<td>312</td>
<td>0</td>
</tr>
<tr>
<td><strong>total</strong></td>
<td>*<em>3410</em></td>
<td>*<em>298</em></td>
</tr>
</tbody>
</table>

* including top-level forms
How does it work -
at compile and chunk time

- LSoar compiles SP’s into Lisp defuns
- Lisp compiles defuns
- Rule-names are stored on attributes
## How does it work - basic structures

<table>
<thead>
<tr>
<th>Working Memory Element</th>
<th>Instantiation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-stamp</td>
<td>Time-stamp</td>
</tr>
<tr>
<td>Node</td>
<td>Rule-name</td>
</tr>
<tr>
<td>Attribute</td>
<td>Bound-lhs-variables</td>
</tr>
<tr>
<td>Value</td>
<td>Supported-by</td>
</tr>
<tr>
<td>Supports</td>
<td>Additions</td>
</tr>
<tr>
<td>Home-base-instantiation</td>
<td>Removals</td>
</tr>
<tr>
<td>Has-i-support</td>
<td>Is-alive</td>
</tr>
<tr>
<td>Visited</td>
<td>Negated-tests</td>
</tr>
<tr>
<td></td>
<td>Negated-conditions</td>
</tr>
<tr>
<td></td>
<td>Has-i-support</td>
</tr>
</tbody>
</table>
How does it work - wme representation

> (spr 'top-state' 3)
(\top-state \^proposed-operator <n-1> <n-3> ^value een ^superstate none )
  (<n-3> ^name increment ^preference <n-2> )
  (<n-2> ^state <top-state> ^accept t)
  (<n-1> ^name decrement ^preference <n-0> )
  (<n-0> ^state <top-state> ^accept t)

> (pprint (symbol-plist 'top-state))
(proposed-operator ([10 : top-state] proposed-operator <n-1> [14 : top-state]
  proposed-operator <n-3>)) value
  ([3 : top-state] value een) superstate ([1 : top-state] superstate none))

> (pprint (symbol-plist <n-1>))
(name ([5 : <n-1> name decrement]) preference ([9 : <n-1> preference <n-0>])))
How does it work -
an example SP

(sp apply-operator
  (state <s> ^value one ^operator <o>)
  (<o> ^name increment)
  -->
  (state <s> ^value one - twee +))
How does it work - an example defun

(DEFUN APPLY-OPERATOR ()
  (LET (-WME1- -WME2- -WME3- <O> <S>)
    (DOLIST (<S> *STATES*)
      (WHEN (SETF -WME1- (TEST-WME <S> 'VALUE 'EEN))
        (DOLIST (-WME2- (GET <S> 'OPERATOR))
          (SETF <O> (WME-VALUE -WME2-))
          (WHEN (SETF -WME3- (TEST-WME <O> 'NAME 'INCREMENT))
            (LET ((VARS (LIST <O> <S>)))
              (WHEN (NOT-EXISTS-SOLUTION 'APPLY-OPERATOR VARS)
                (LET (INST (WME-LIST (LIST -WME1- -WME2- -WME3-)))
                  (SETF INST (MAKE-INSTANTIATION :RULE-NAME 'APPLY-OPERATOR
                                                :VARIABLES VARS :SUPPORTED-BY WME-LIST))
                  (PUSH INST (GET 'APPLY-OPERATOR 'NEW-INSTANTIATIONS))
                  (PUSH INST (GET 'APPLY-OPERATOR 'NEW-INSTANTIATIONS))
                  (PUSHNEW 'APPLY-OPERATOR *RULES-THAT-FIRED*)
                  (PROPOSE-ADD <S> 'VALUE 'TWEE INST NIL)
                  (PROPOSE-DELETE <S> 'VALUE 'EEN INST))))))))
How does it work - an example defun

(DEFUN APPLY-OPERATOR ()
  (LET (-WME1- -WME2- WME3- <O> <S>)
    (DOLIST (<S> *STATES*)
      (WHEN-TEST-WME -WME1- <S> 'VALUE 'EEN)
        (DOLIST-VAL (-WME2- (GET <S> 'OPERATOR) <O>)
          (WHEN-TEST-WME WME3- <O> 'NAME 'INCREMENT)
            (WHEN (NOT-EXISTS-SOLUTION 'APPLY-OPERATOR <O> <S>)
              (LET ((VARS (LIST <O> <S>)) INST (WME-LIST (LIST -WME1- -WME2- -WME3-)))
                (SETF INST (MAKE-INST 'APPLY-OPERATOR VARS WME-LIST)
                  (PROPOSE-ADD <S> 'VALUE 'TWEE INST NIL)
                  (PROPOSE-DELETE <S> 'VALUE 'EEN INST))))))))))
How does it work - instantiations

> (pprint (symbol-plist 'propose-operator))
(NEW-INSTANTIATIONS NIL
INSTANTIATIONS

  Instantiation from rule: PROPOSE-OPERATOR
  variables: ([TOP-STATE])
  supported by: ([3 : <TOP-STATE> VALUE EEN])
  additions:
([14 : <TOP-STATE> PROPOSED-OPERATOR <N-3>] [13 : <N-3> PREFERENCE <N-2>] [12 : <N-2> ACCEPT T] [11 : <N-2> STATE <TOP-STATE>] [10 : <TOP-STATE> PROPOSED-OPERATOR <N-1>] [9 : <N-1> PREFERENCE <N-0>] [8 : <N-0> ACCEPT T] [7 : <N-0> STATE <TOP-STATE>] [6 : <N-3> NAME INCREMENT] [5 : <N-1> NAME DECREMENT])
  removals: NIL)
How does it work - run-time cycle

(defun d (&optional (num 1))
 (dotimes (i num)
   (when *soar-input-functions*
     (try-input-functions)
     (when *instantiations-to-remove*
       (retract))
    (dolist (rule *potential-rules*)
      (funcall rule))
   (setf *potential-rules* nil)
   (handle-rules-that-fired) ; and setf *rules-that-fired* nil
   (when (or *result-add-wmes* *result-delete-wmes*)
     (make-all-chunks))
   (handle-attributes-negtests)
   (handle-attributes-negcons)
   (when *instantiations-to-remove*
     (retract))
   (try-output-functions)
   (when (not *potential-rules*) ; quiescence
     (main-decide)
     (retract))))
(init-soar)(d 4)

--- Memory phase (0 : 0) -----------

=> [3 : <TOP-STATE> VALUE EEN] INIT-STATE (1)

--- Memory phase (0 : 1) -----------

=> [14 : <TOP-STATE> PROPOSED-OPERATOR <N-3>] PROPOSE-OPERATOR (3)
=> [13 : <N-3> PREFERENCE <N-2>] PROPOSE-OPERATOR (3)
=> [12 : <N-2> ACCEPT T] PROPOSE-OPERATOR (3)
=> [10 : <TOP-STATE> PROPOSED-OPERATOR <N-1>] PROPOSE-OPERATOR (3)
=> [9 : <N-1> PREFERENCE <N-0>] PROPOSE-OPERATOR (3)
=> [8 : <N-0> ACCEPT T] PROPOSE-OPERATOR (3)
=> [7 : <N-0> STATE <TOP-STATE>] PROPOSE-OPERATOR (3)
=> [6 : <N-3> NAME INCREMENT] PROPOSE-OPERATOR (3)
=> [5 : <N-1> NAME DECREMENT] PROPOSE-OPERATOR (3)

--- Memory phase (0 : 2) -----------

--- Decide (0 : 3) ------------------

=> [16 : <TOP-STATE> SUBSTATE <N-4>] @@DECIDE-NO-TRACE@@ NIL
=> [17 : <N-4> ATTRIBUTE OPERATOR] @@DECIDE-NO-TRACE@@ NIL
=> [18 : <N-4> CHOICES MULTIPLE] @@DECIDE-NO-TRACE@@ NIL
=> [19 : <N-4> IMPASSE TIE] @@DECIDE-NO-TRACE@@ NIL
=> [21 : <N-4> ITEM <N-3>] @@DECIDE-TRACE@@ (14)
=> [23 : <N-4> ITEM <N-1>] @@DECIDE-TRACE@@ (10)
=> [24 : <N-4> SUPERSTATE <TOP-STATE>] @@DECIDE-NO-TRACE@@ NIL

--- Memory phase (1 : 0) -----------

=> [28 : <N-3> PREFERENCE <N-5>] RESOLVE-TIE (24 17 19 21 6 23 5)
=> [27 : <N-5> BINARY-BETTER <N-1>] RESOLVE-TIE (24 17 19 21 6 23 5)
=> [26 : <N-5> STATE <TOP-STATE>] RESOLVE-TIE (24 17 19 21 6 23 5)

--- Chunking phase (1 : 0) -----------

Building CHUNK-2