

ACT-R: Learning the Architecture, Creating, and Debugging Models

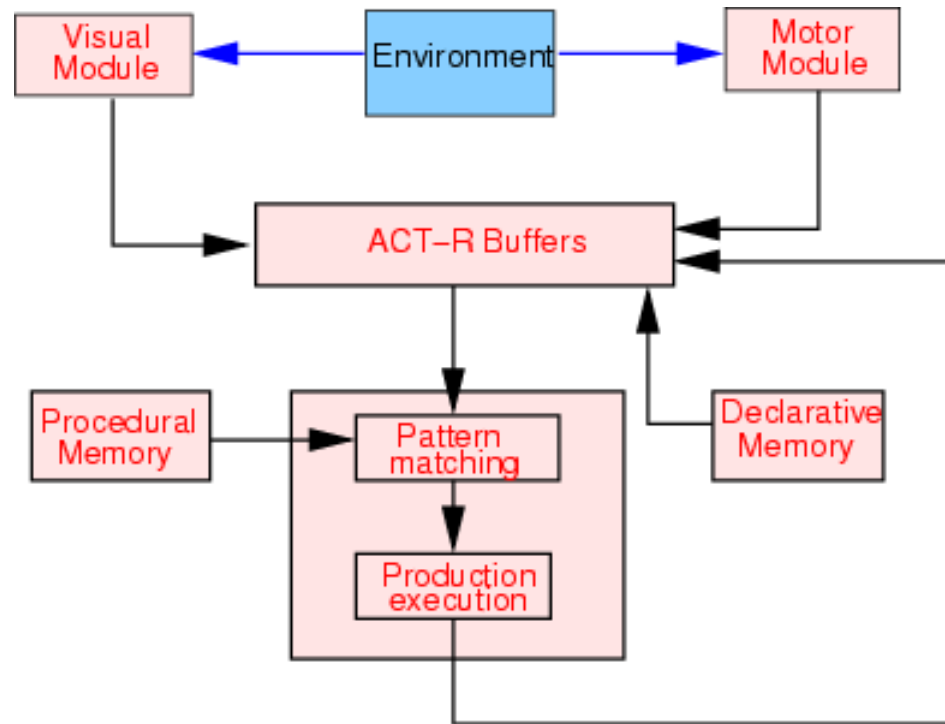
Bradley Best (bbest@andrew.cmu.edu)

Human Computer Interaction Institute

Carnegie Mellon University

Before We go Farther, Time for ACT-R 101

- Declarative Memory
 - a.k.a. Working memory, chunks, wmes
- Procedural Memory
 - a.k.a. Productions, condition-action pairs
- Production Cycle
 - Construct conflict set from matching productions, select best match from conflict set, fire instantiation



Chunks and Productions

Chunks:

```
Action023:  
  isa chase  
  agent dog  
  object cat  
Fact3+4:  
  isa addition-fact  
  addend1 three  
  addend2 four  
  sum seven
```

Productions:

```
IF the goal is to classify a person  
  and he is unmarried  
THEN classify him as a bachelor  
  
IF the goal is to add two digits d1 and d2 in a column  
  and  $d1 + d2 = d3$   
THEN set as a subgoal to write d3 in the column
```

Back on Topic: Learning ACT-R

- Online tutorial
- Learning by example
 - Published models
- Learning through instruction
 - Workshops
 - University classes
- User Community
 - ACT-R mailing list
- Going to the well:
 - Christian Lebiere, John Anderson, Jon Fincham, Dan Bothell, Scott Douglass...

Online Tutorial

- Contents:

Unit1: Understanding Production Systems

Unit2: Perception and Motor Actions in ACT-R
experiment descriptions

Unit3: Attention
experiment descriptions

Unit4: Complex Processing
experiment descriptions

Unit5: Activation and Latency
experiment descriptions

Unit6: Activation and Probability of Recall
experiment descriptions

Unit7: Base-level Learning and Accuracy
experiment descriptions

Unit8: Selecting Productions on the Basis of Their Utilities and Learning these Utilities
experiment descriptions

Unit9: Production Rule Learning

Tutorial Excerpt:

4. Click the **Run** button in the **Control Panel** to start the assignment. The first production to apply, **Start**, will appear in the **Stepper**. Its structure will be displayed in the lower right pane of the window and all of the variables will be highlighted. Your task is to go through the production rule replacing all the variables with the values to which they are bound. When you click on a variable a dialog will open in which you can enter its value. Here are the rules for doing this:

- **=goal** will always bind to the current contents of the goal buffer. This can be found in the **Buffer window**.
- **=retrieval** will always bind to the chunk in the retrieval buffer. This can also be found in the **Buffer window**.
- If a variable has been bound to a value, it must be assigned the same value throughout the matching. The bound values are displayed in the middle right pane of the **Stepper**.
- At any point in time, you can ask the tutor for help in binding a variable by hitting either the **Hint** or **Help** button of the entry dialog. A hint will instruct you on where to find the correct answer and help will give you the answer.

6. Once the production is completely instantiated, you can fire it by hitting the **Step** button at the top of the **Stepper** window. The **Stepper** will then stop at the firing of that production and after you step past that you will see the retrieval event for the count-order chunk C.

Published models (code, data, sim)

- Anderson, J. R. & Douglass, S. (2001). Tower of Hanoi: Evidence for the Cost of Goal Retrieval. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 27(6). [] [[ACT-R 4.0 web-based simulations, model source code and experimental data spreadsheet](#)] [[info](#)]
- Anderson, J. R. & Douglass, S. (unpublished). Visual Attention and Problem Solving. [[Excel files](#)] [[info](#)]
- Anderson, J. R. & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Erlbaum. [[Chapter abstracts, model source code, and web-based simulations](#)] [[info](#)]
- Anderson, J. R. & Matessa, M. P. (1997). A production system theory of serial memory. *Psychological Review*, 104, 728-748. [] [[Excel files and LISP code](#)] [[info](#)]
- Anderson, J. R. & Reder, L. M. (1999). The fan effect: New results and new theories. *Journal of Experimental Psychology: General*, 128, 186-197. [] [[ACT-R 4.0 web-based simulations](#)] [[info](#)]
- Anderson, J. R., & Betz, J. (2001). A hybrid model of categorization. *Psychonomic Bulletin and Review*, 8, 629-647. [] [[ACT-R 4.0 web-based simulation and model source code](#)] [[info](#)]
- Anderson, J. R., Bothell, D., Lebiere, C. & Matessa, M. (1998). An integrated theory of list memory. *Journal of Memory and Language*, 38, 341-380. [] [[ACT-R 4.0 web-based simulations](#)] [[info](#)]
- Anderson, J. R., Budson, R. & Reder, L. M. (2001). A theory of sentence memory as part of a general theory of memory. *Journal of Memory and Language*, 45, 337-367. [] [[ACT-R 4.0 web-based simulations and model source code](#)] [[info](#)]
- Anderson, J. R., Fincham, J. M. & Douglass, S. (1999). Practice and retention: A unifying analysis. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 25, 1120-1136. [] [[Excel files](#)] [[info](#)]
- Anderson, J. R., Matessa, M., & Lebiere, C. (1997). ACT-R: A theory of higher level cognition and its relation to visual attention. *Human Computer Interaction*, 12(4), 439-462. [] [[Excel files](#)] [[info](#)]
- Anderson, J. R., Qin, Y., Sohn, M.-H., Stenger, V. A. & Carter, C. S. (in press). An information-processing model of the BOLD response in symbol manipulation tasks. *Psychonomic Bulletin and Review*. [] [[ACT-R 5.0 web-based simulations and model source code](#)] [[info](#)]
- Budson, R. & Anderson, J. R. (submitted). Interpretation-Based Processing: A Unified Theory of Semantic Sentence Processing. [] [[ACT-R 4.0 web-based simulations and model source code](#)] [[info](#)]
- Budson, R. and Anderson, J. R. (2001). *Word Learning in Context: Metaphors and Neologisms*. (Technical Report No. CMU-CS-01-147.) School of Computer Science, Carnegie Mellon University. [] [[ACT-R 4.0 web-based simulation and Excel files](#)] [[info](#)]
- Gluck, K. A., Lovett, M. C., Anderson, J. R., & Shute, V. J. (unpublished). The Curriculum and the Interface: A Componential Analysis of the Learning Curve. [[ACT-R 4.0 web-based simulation and model source code](#)] [[info](#)]
- Gunzelmann, G., & Anderson, J. R. (in press). Problem solving: Increased planning with practice. *Cognitive Systems Research*. [[ACT-R 4.0 web-based simulation and model source code](#)] [[info](#)]
- Gunzelmann, G., & Anderson, J. R. (in press). Strategic differences in the coordination of different views of space. Paper and Talk to be presented at the 24th Annual Meeting of the Cognitive Science Society, Fairfax, VA: August, 2002. [] [[ACT-R 5.0 model source code](#)] [[info](#)]
- Gunzelmann, G., Anderson, J. R., & Douglass, S. (submitted). Orientation tasks with multiple views of space: Strategy variability and performance. *Cognitive Psychology*. [] [[ACT-R 5.0 web-based simulation, model source code, and abstract](#)] [[info](#)]
- Lebiere, C., & West, R. L. (1999). A dynamic ACT-R model of simple games. In *Proceedings of the Twenty-first Conference of the Cognitive Science Society*, pp. 296-301. Mahwah, NJ: Erlbaum. [[ACT-R 4.0 web-based simulation and model source code](#)] [[info](#)]
- Matessa, M. P. (unpublished). Communication in collaborative problem solving. [[ACT-R 4.0 model source code](#)] [[info](#)]
- Matessa, M., & Anderson, J. R. (2000). Modeling Focused Learning in Role Assignment. *Language and Cognitive Processes*, 15 (3), 263-292. [] [[ACT-R 4.0 web-based simulations and models source code](#)] [[info](#)]
- Murdock, J. W., Simina, M., Davies, J., & Shippey, G. (1998). Modeling Invention by Analogy in ACT-R. In *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society* (pp. 740-745). Mahwah, NJ: Lawrence Erlbaum Associates. [[ACT-R model source code and postscript paper](#)] [[info](#)]
- Pavlik, P. I. and Anderson, J. R. (submitted). Practice and Forgetting Effects on Vocabulary Memory: An Activation Based Model of the Spacing Effect. [[Model and Sequence Files](#)] [[info](#)]
- Peterson, S. & Simon, T. (2000). Computational Evidence for the Subitizing Phenomenon as an Emergent Property of the Human Cognitive Architecture. *Cognitive Science*, 24(1). [[ACT-R 3.0 model source code](#)] [[info](#)]
- Salvucci, D. D., & Anderson, J. R. (2001). Integrating analogical mapping and general problem solving: The path-mapping theory. *Cognitive Science*, 25, 67-110. [] [[ACT-R 4.0 web-based simulations and models source code](#)] [[info](#)]
- Sohn, M.-H., & Anderson, J. R. (2001). Task preparation and task repetition: Two-component model of task switching. *Journal of Experimental Psychology: General*. [[ACT-R 4.0 web-based simulations and model source code](#)] [[info](#)]
- Taatgen, N. A., & Anderson, J. R. (submitted). Why do children learn to say "Broke"? A model of learning the past tense without feedback. *Cognition*. [] [[ACT-R 4.0 model source code](#)] [[info](#)]

Workshops

The ACT-R community gathers every summer for two events: the annual ACT-R summer school and workshop.



- Upcoming
 - Tenth Annual Workshop and Summer School July 2003
- Past Events
 - Ninth Annual Workshop and Summer School July/August 2002
 - Eighth Annual Post-Graduate Summer School July 2001
 - Seventh Annual Workshop and Summer School July/August 2000
 - Sixth Annual Workshop August 1999
 - Sixth Annual Summer School July/August 1999
 - Fifth Annual Workshop and Summer School July 1998
 - Fourth Workshop and Summer School August 1997
 - Third Workshop and Summer School June 1996

User Community

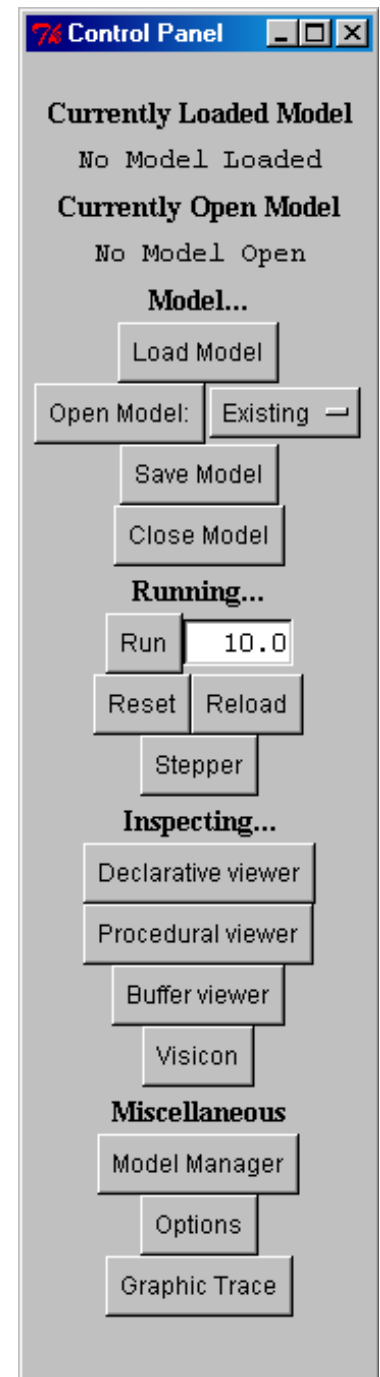
- ACT-R mailing list
 - Ask a question, get an answer (or two, or three, or four)
- Active community
 - ~ 100 Published Models in ACT-R 1997-2002
 - Several universities offering classes in ACT-R
 - Military, Industrial, and Academic users

Creating Models

- Use structured editor in environment
 - Procedural memory
 - Graphical access to symbolic and sub-symbolic
 - Declarative memory
 - Graphical access to symbolic and sub-symbolic
 - Global parameter setting
- Use standalone editors
 - Look Ma, no hands!

Modeling in the Environment

- The ACT-R environment supports graphical access to the key features of the architecture
 - Supports creating, running, debugging, and management of model files



Standalone Model Creation

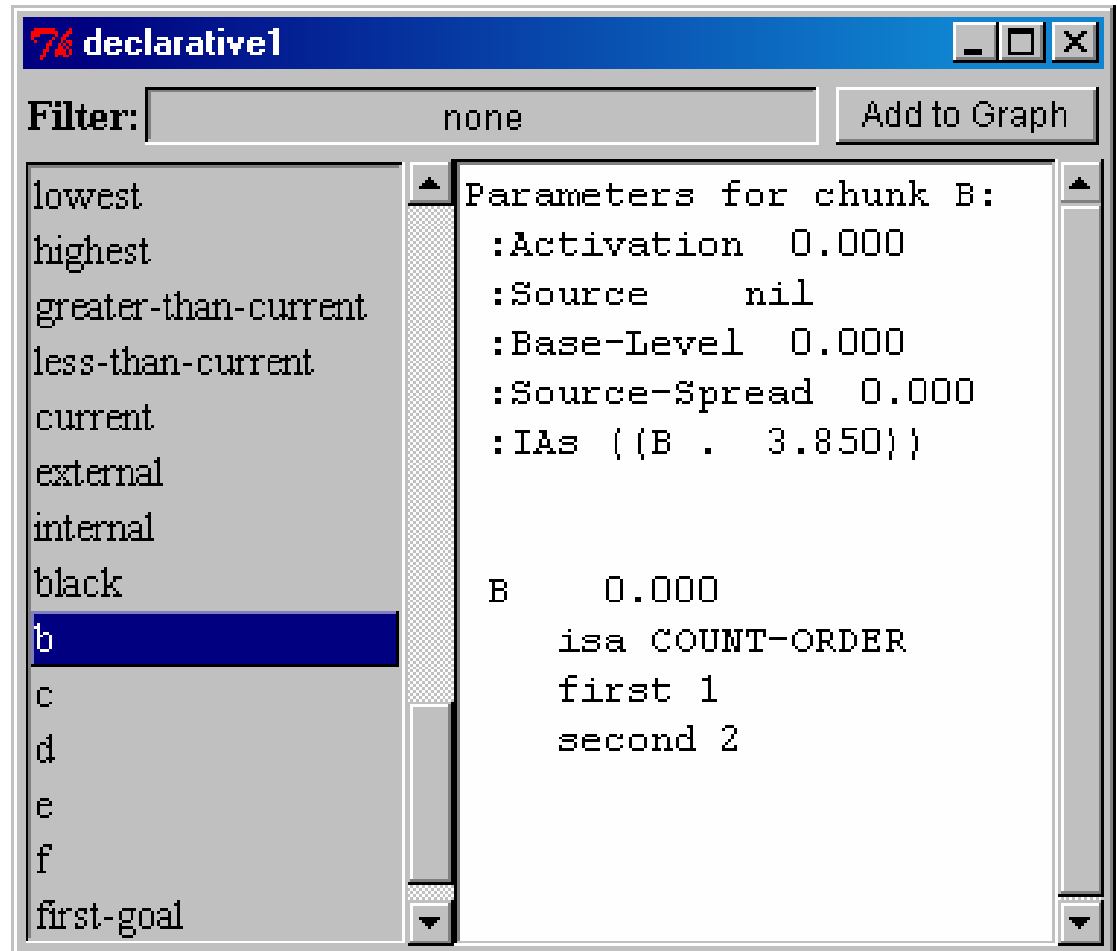
- Pick your favorite text editor
 - Emacs?
- Type in model and save file
- Start Lisp
- Load act-r5.lisp
- Load model file
- Clear, and run model

Debugging Models

- Within the structured editor
 - Inspecting the contents of declarative memory
 - Inspecting the contents of procedural memory
 - Tracing a model
 - Why won't this production fire?
- Standalone debugging (into the guts)
 - Look Ma, no hands and no feet!

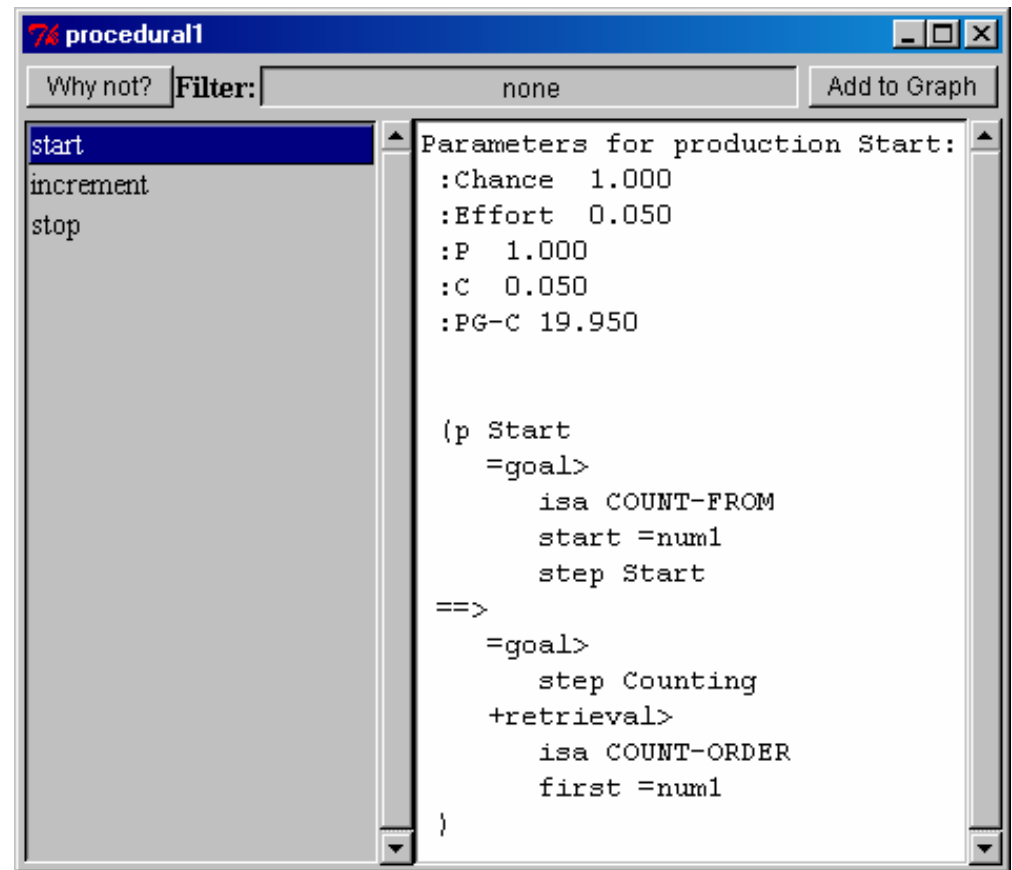
Inspecting Declarative Memory

- On the left side is a list of chunks
- On the right is the display of the currently selected chunk from the list.



Inspecting Procedural Memory

- On the left side is a list of productions
- On the right is the display of the currently selected production from the list.

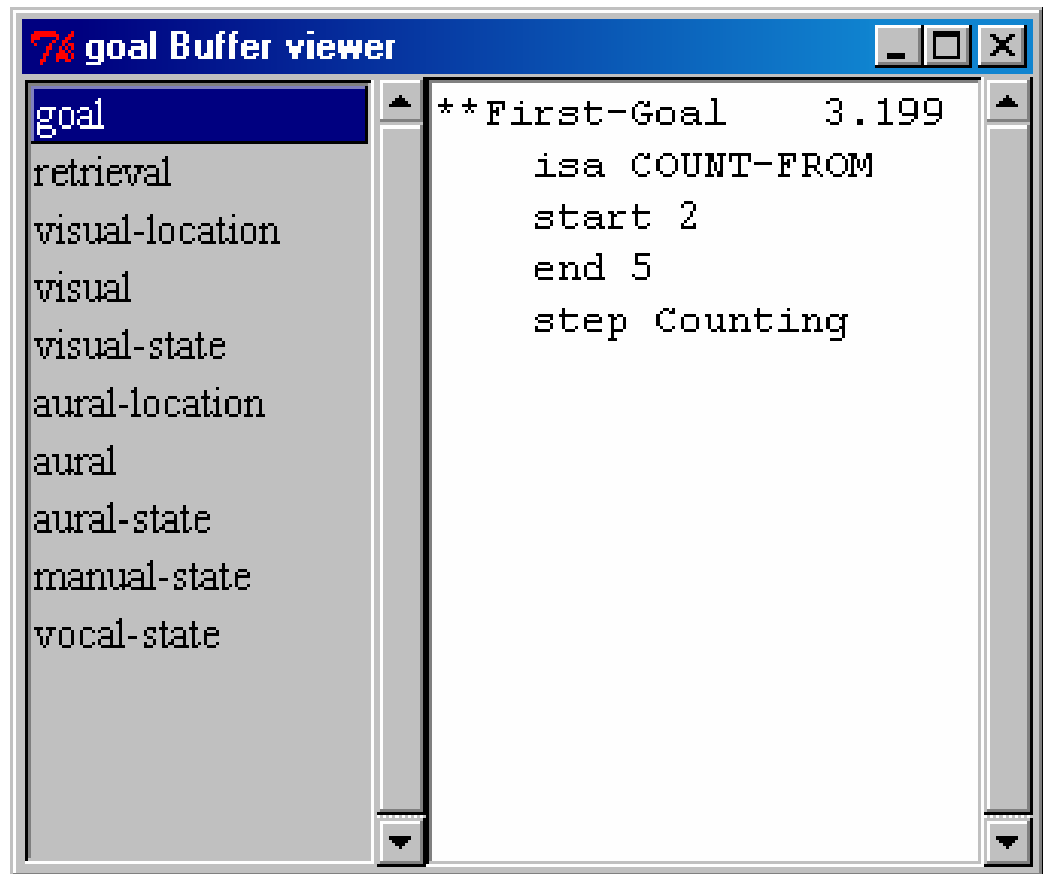


The screenshot shows a window titled '74 procedural1'. At the top, there is a 'Filter:' dropdown menu set to 'none' and an 'Add to Graph' button. The left pane contains a list of productions: 'start', 'increment', and 'stop'. The 'start' production is selected. The right pane displays the parameters for the selected production:

```
Parameters for production Start:  
:Chance 1.000  
:Effort 0.050  
:P 1.000  
:C 0.050  
:PG-C 19.950  
  
(p Start  
 =goal>  
   isa COUNT-FROM  
   start =num1  
   step Start  
==>  
 =goal>  
   step Counting  
+retrieval>  
   isa COUNT-ORDER  
   first =num1  
)
```

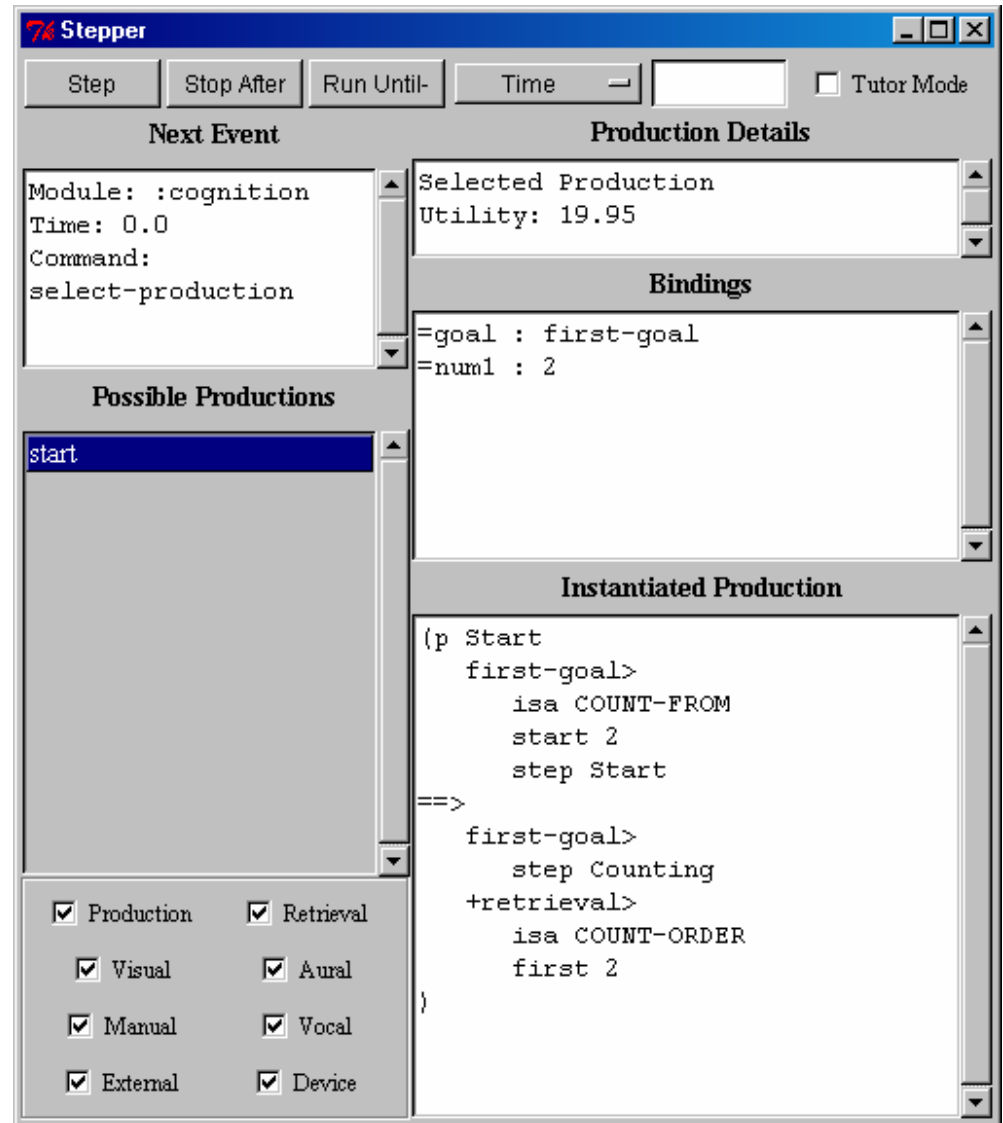
Inspecting Buffer Contents

- On the left is a list of the buffers in ACT-R and on the right is displayed the current contents of the selected buffer.



Using the Stepper to Inspect a Production Event

- The stepper dialog is used to “step” an ACT-R model through its execution one “event” at a time.
- When it is open it will stop the model at all of the requested points, and wait for you to tell it to continue.
- A production event can be generated on either a production selection (select-production command) or a production firing (execute-rhs command).



Using the Stepper to Inspect a Retrieval

- When a chunk is retrieved it will generate an event with the complete-retrieval command.

The screenshot shows the Stepper software interface with the following components:

- Control Panel:** Buttons for "Step", "Stop After", "Run Until-", and "Time" (with a dropdown arrow). A "Tutor Mode" checkbox is also present.
- Next Event:** A text area displaying:

```
Module: :cognition
Time: 0.1
Command:
complete-retrieval
0.1
```
- Possible Chunks:** A list of chunks labeled 'c', 'b', 'd', 'e', and 'f'. Chunk 'c' is highlighted in blue.
- Retrieved Chunk:** A text area displaying:

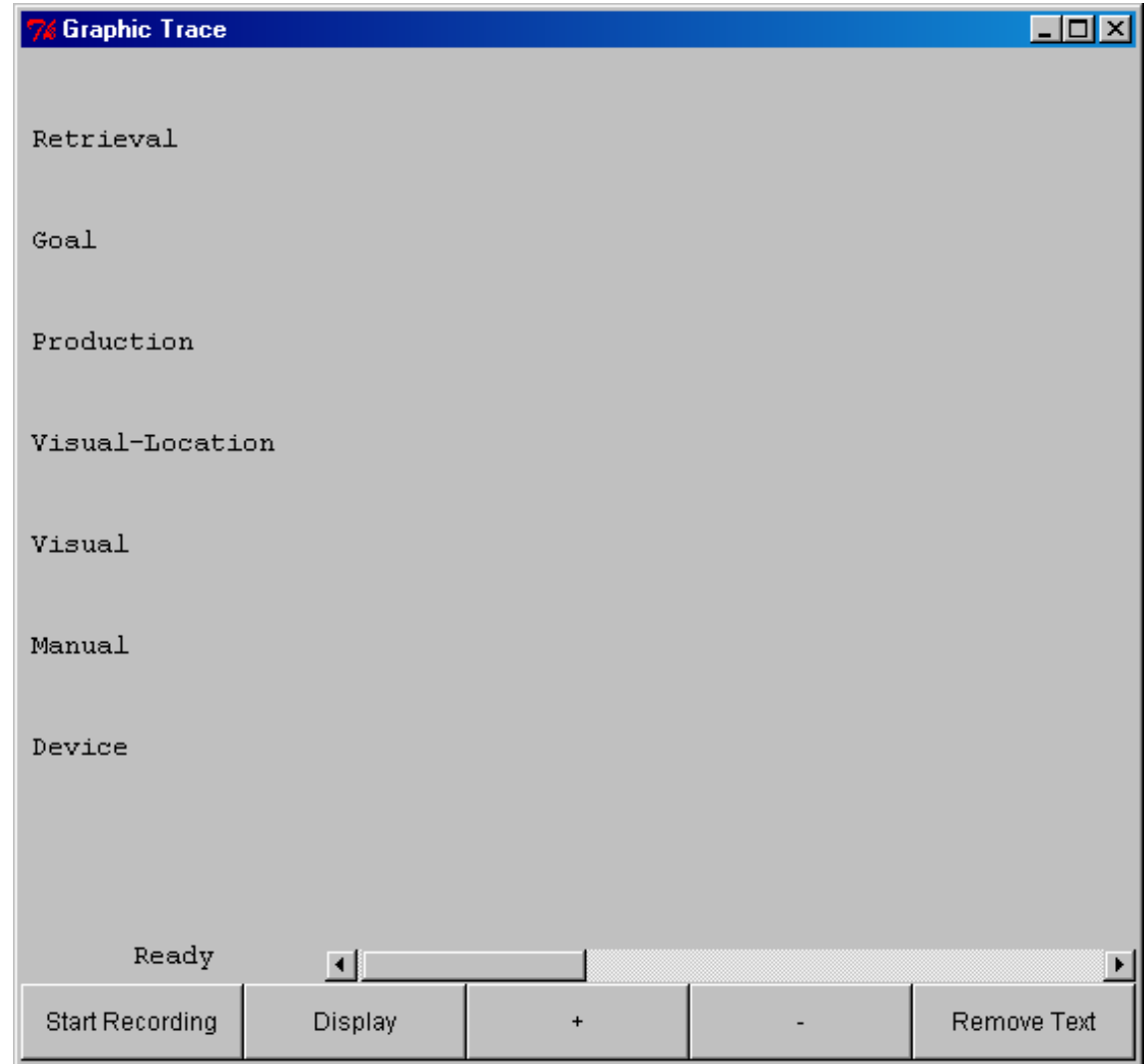
```
Retrieved Chunk
Matching Score: 0.0
```
- Retrieval Request:** A text area displaying:

```
Sources: (Start)
Retrieval Request:
+retrieval>
  isa COUNT-ORDER
  first 2
```
- Chunk:** A text area displaying:

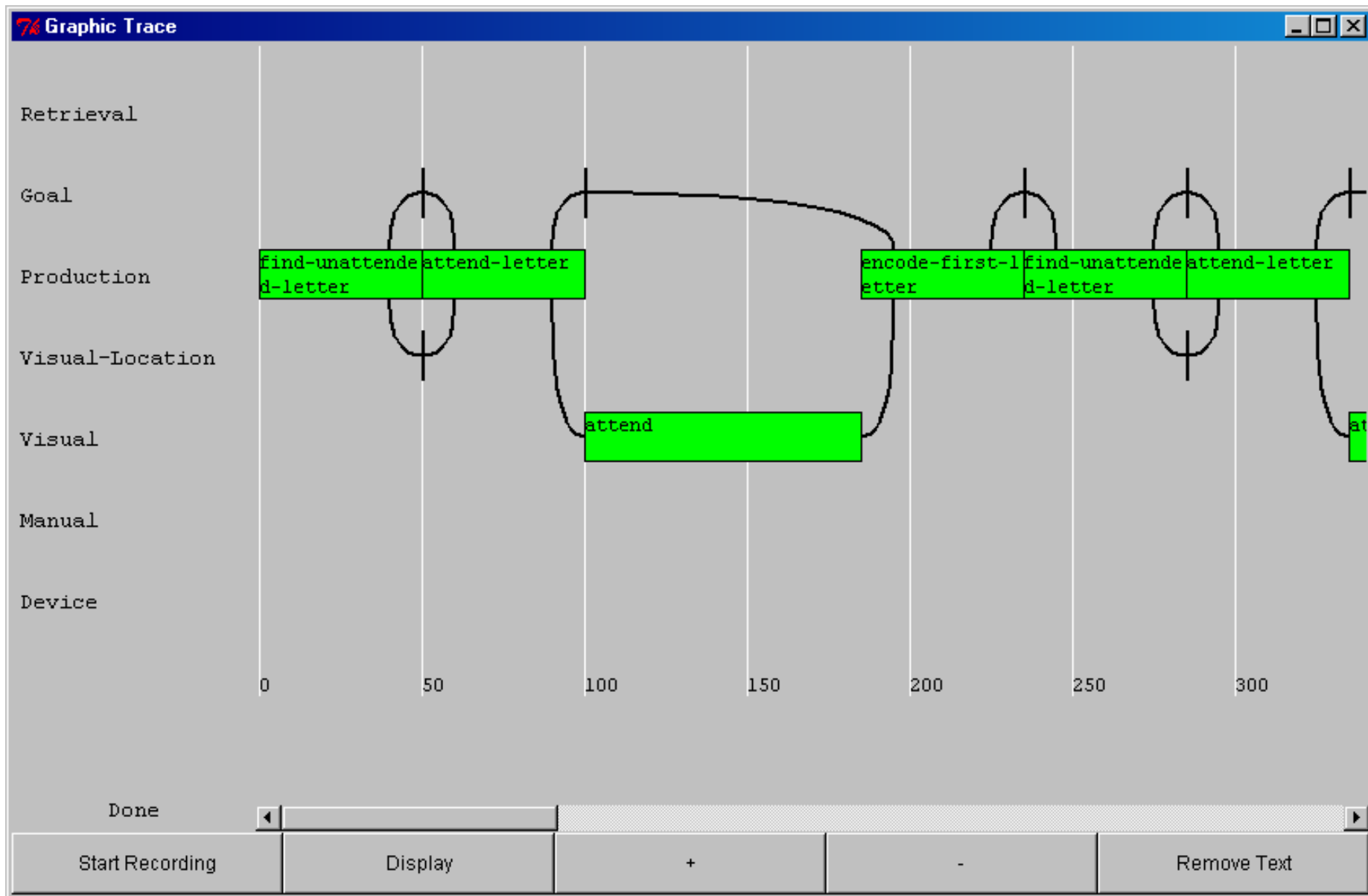
```
c 0.000
  isa COUNT-ORDER
  first 2
  second 3
```
- Checkboxes:** A grid of checked checkboxes for "Production", "Retrieval", "Visual", "Aural", "Manual", "Vocal", "External", and "Device".

Graphic Trace of Model Execution

- Graphic trace of the dependencies of the modules for a run (like a PERT chart).
- Each of the main modules of ACT-R 5 is represented on a row of the diagram
- Time advances along the horizontal axis
- A box on a row (other than the Production's row) indicates that that module has had a request made to it and it is busy

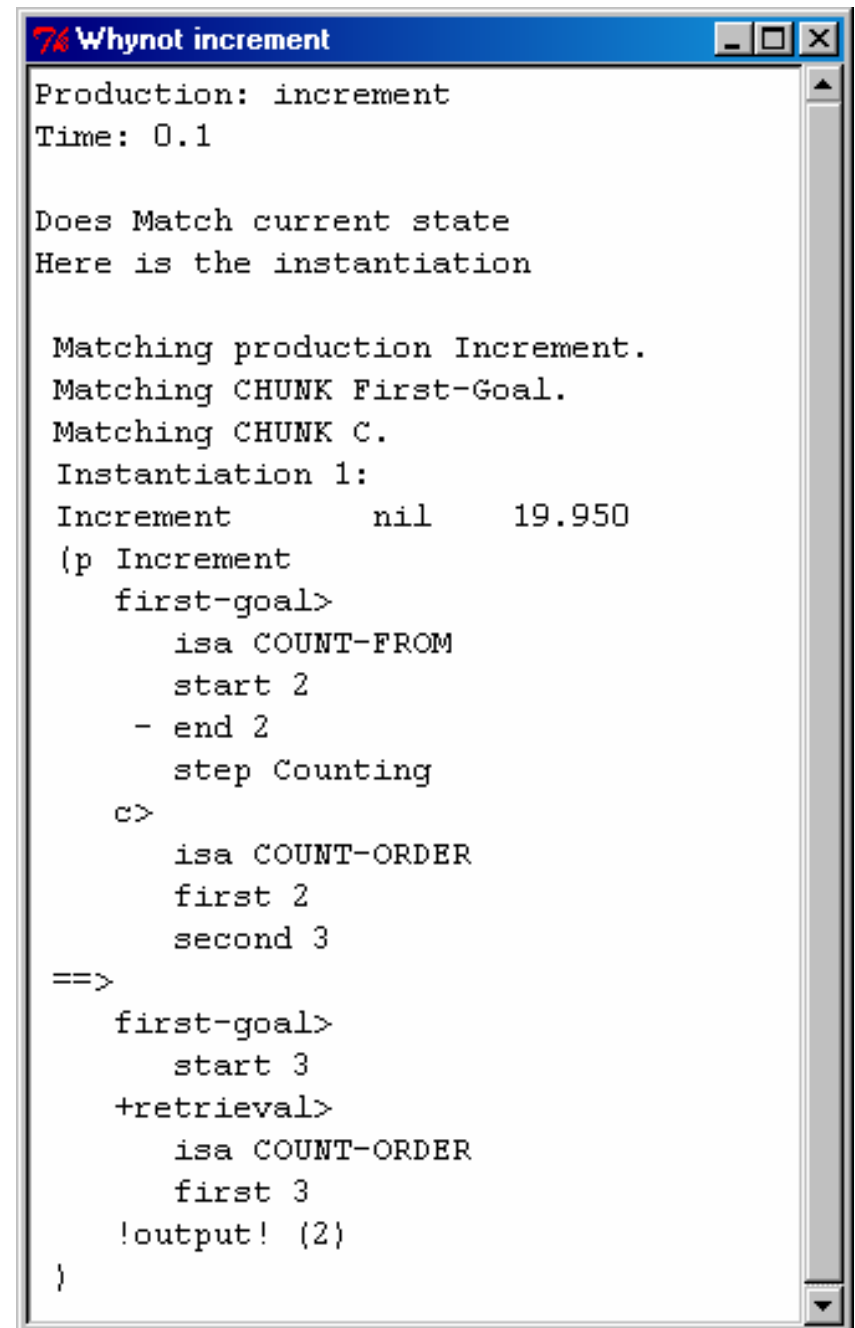


Graphic Trace During a Run



Why Will this Production Fire?

- The “Why not?” button is used to call the ACT-R whynot function. That will open a new window which displays whether the current production matches the current buffer contents or not and if so what its instantiation is and if not why it does not match.



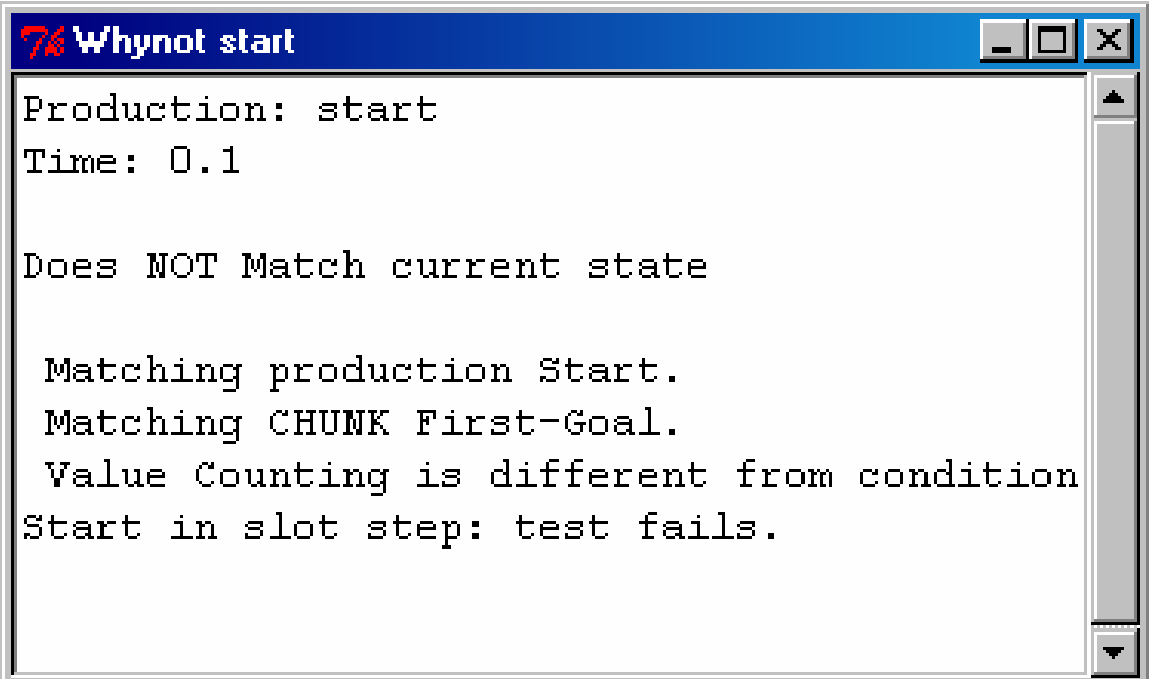
```
7% Why not increment
Production: increment
Time: 0.1

Does Match current state
Here is the instantiation

Matching production Increment.
Matching CHUNK First-Goal.
Matching CHUNK C.
Instantiation 1:
Increment      nil      19.950
(p Increment
  first-goal>
    isa COUNT-FROM
    start 2
  - end 2
    step Counting
  c>
    isa COUNT-ORDER
    first 2
    second 3
==>
  first-goal>
    start 3
  +retrieval>
    isa COUNT-ORDER
    first 3
  !output! (2)
}
```

Why Won't this Production Fire?

- The production selected does not match the current state. The value 'Counting' is different from condition 'Start' in slot step.



```
76 Whynot start
Production: start
Time: 0.1

Does NOT Match current state

Matching production Start.
Matching CHUNK First-Goal.
Value Counting is different from condition
Start in slot step: test fails.
```

Standalone Debugging: Into the Guts

- Inspecting the contents of declarative memory
 - (dm)
 - (dm <chunk-name>)
- Inspecting the contents of procedural memory
 - (pp)
 - (pp <production-name>)
- Tracing a model
 - (run 1)
 - (pmatches)
- Why won't this production fire?
 - (whynot <production-name>)

Gold Nuggets

- Lots of detail available at millisecond level
- Environment runs on Mac, PC, others
- Helpful user community
- Good online resources
- Plenty of examples (published models) to work from
- Source code is very open (it's Lisp: the source code executes)

Lumps of Coal

- Experienced programmers always seem to end up using a text editor and Lisp
- Understanding sub-symbolic computations in ACT-R is still a challenge
- Lacking facilities for managing large sets of productions and planning their interactions
- No built-in version control

Questions?