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:
- **Unit 1:** Understanding Production Systems
- **Unit 2:** Perception and Motor Actions in ACT-R experiment descriptions
- **Unit 3:** Attention experiment descriptions
- **Unit 4:** Complex Processing experiment descriptions
- **Unit 5:** Activation and Latency experiment descriptions
- **Unit 6:** Activation and Probability of Recall experiment descriptions
- **Unit 7:** Base-level Learning and Accuracy experiment descriptions
- **Unit 8:** Selecting Productions on the Basis of Their Utilities and Learning these Utilities experiment descriptions
- **Unit 9:** 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)


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.
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.
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.
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.
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?