ACT-R Updates Summer 2014

Dan Bothell
Carnegie Mellon University
db30@andrew.cmu.edu
New Website
ACT-R 6.0 Additions
simulate-retrieval-request command

> (simulate-retrieval-request isa count-order first 3)
Chunk J does not match
Chunk I does not match
Chunk H does not match
Chunk G does not match
Chunk F does not match
Chunk E does not match
Chunk D matches
Chunk C does not match
Chunk B does not match
Chunk A does not match
Chunk D has the current best activation 0.0
Chunk D with activation 0.0 is the best
(D)
Style Warnings

• Additional inter-production checks
  – Conditions with types or slots not set in actions or initial state
  – Actions which modify slots that aren’t tested

• :style-warnings set to nil will suppress them
Starting parameters

- New system parameter called :starting-parameters

- Set to a list of parameters and values which are appropriate for use with sgp

  (ssp :starting-parameters (:esc t :trace-detail high))

- Those settings applied at the start of every model definition and at the beginning of every reset
Motor module extensions

• Collection of motor addons available in extras
  – holding and releasing actions for keys
  – buffers for tracking hands and fingers individually
  – more high-level 'press-key like' actions
Multithreaded calculations

• Speculative code available in extras
• Take advantage of multiple cores in a machine
  • Find-matching-chunks
  • Compute activations
  • Perform blending
Android ACT-R Environment app

• Full ACT-R Environment as an Android app
• Get it from the ACT-R website
• Possible because of AndroWish Tcl/Tk system
• There is not a Tcl/Tk system available for iOS
ACT-R 6.0 Issues
• Incongruence between chunk-types and dynamic productions’ abilities to extend chunks and to variablize conditions and actions

• Issues with non-merging of apparently similar chunks
  – Overloaded use of nil

• Inability for productions to detect some states
  – Both “slot t” and “- slot t” false
Static chunk-types

- Discussed at last year’s workshop
- Create almost as many issues as they solve
ACT-R 6.1

The Chunk-type is dead, long live the Chunk-type.
Why not just fix 6.0?

• The right fix seems to be a conceptual change
• Could break existing 6.0 models
• Allows for “fixing” other things as well
Chunks do not have a type!

- A chunk is a collection of slots and non-nil values
- A slot value of nil means that the chunk does not have the slot
  - Both for setting slot values and testing them

<table>
<thead>
<tr>
<th>6.0</th>
<th>6.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHUNK</td>
<td>CHUNK</td>
</tr>
<tr>
<td>ISA TEST</td>
<td>SLOT1 &quot;value&quot;</td>
</tr>
<tr>
<td>SLOT1 &quot;value&quot;</td>
<td></td>
</tr>
<tr>
<td>SLOT2 NIL</td>
<td>slot1 slot3</td>
</tr>
<tr>
<td>SLOT3 NIL</td>
<td>slot1 =s</td>
</tr>
<tr>
<td></td>
<td>=goal&gt;</td>
</tr>
<tr>
<td></td>
<td>isa test</td>
</tr>
<tr>
<td></td>
<td>=s nil</td>
</tr>
</tbody>
</table>

==>
Don’t eliminate chunk-types

• Useful tool for the modeler
• Allow chunk-type creation and isa like before
• Don’t require that isa be used anywhere

• Not used by the model!
• Important differences in 6.1 for isa
  – NOT a test in a production condition
  – NOT a component of a request to a module
Make chunk-type more useful in new role

- Allow multiple inheritance

- Default chunk-type slot value expansion in both chunk and production definitions
(define-model example
  (sgp :v t))

(chunk-type example (slot t))

(define-chunks
  (example isa example))

(pprint-chunks example)

(p e1
  ?goal>
    buffer empty
  ==>
  +goal>
    isa example)

(p e2
  =goal>
    isa example
  ==>)

(run 1))
Issue with production syntax

• If isa is optional, what about the difference between these
  
  +goal> slot value
  +goal> isa something slot value
New production action indicators

• * is used for modification requests
  previously + without an isa

• @ is for the buffer overwrite action
  previously =buffer> chunk now @buffer> chunk
No special cases in production actions

If type x has no default slots and chunk c looks like this

\[
\begin{array}{l}
\text{c} \\
\text{SLOT} \quad \text{VALUE}
\end{array}
\]

• These all do the same thing
  =goal> isa x slot value, =goal> isa chunk slot value, =goal> slot value, =goal> c

• These also do the same as above (through the goal module)
  *goal> isa x slot value, *goal> isa chunk slot value, *goal> slot value, *goal> c

• These are also all the same (not same as above)
  +goal> isa x slot value, +goal> isa chunk slot value, +goal> slot value, +goal> c
How do the other modules work?

• The information must be in a slot
• For the PM modules all of the chunk-types now have a slot named cmd which has a default value which matches the type name

```
+manual>
  isa press-key
  key "a"
```

```
+manual>
  cmd press-key
  key "a"
```
But I like having my chunks typed

- Could use a slot to hold a type value
- Conceptually, a common type means common underlying structure
- Better to provide common structure in chunks
  - Give the type a unique slot with a default value
    (chunk-type type-a (isa-type-a t)...)
Other changes

• Collapse the p/p* distinction
• Simplify production condition syntax
  – One buffer test and/or one query per buffer
• Cannot modify chunks in DM now
  – The :fast-merge switch is gone
Will my ACT-R 6.0 model work as-is?

• 50 test models with ACT-R 6.0
  – 41 work the same
  – 48 work if the :backwards system parameter set

• 25 of those models are from the tutorial
  – 21 work the same
  – 25 work with :backwards set
Typical issue to fix

• Production conditions/requests or Lisp code which differentiate based on the isa value

```lisp
(p needs-the-isa
  =goal>
    isa task1
  ==>
    +retrieval>
    isa task1-data)
```

• Setting the :backwards switch corrects that
Will require changes to model/code

• Lisp code which tests chunk types
  – Chunk-chunk-type or chunk-spec-chunk-type

• Probably includes any modules which process requests
Status

• The code is ready and available through subversion svn://act-r.psy.cmu.edu/actr6.1

• The tutorial and documentation is not yet updated

• End of year update should make 6.1 primary download
Blending

• No types changes some things

• Blend over all slots which exist in the chunks matching the request
  – Not the slots of the type requested

• Common “type” for possible values now means overlapping slots
  – Intersection of the slots
  – All the chunks in DM with that set of slots
Blending cont.

• Not having nil slot values changes blending scenarios

• Blended value computed from the chunks which have the slot
  – Previously used all chunks and considered values of nil

• When there was any nil value it didn’t really “blend”
  – Case d which picked the best value among them
• Old method would result in either 3.0, 1.0, or nil
• Now it will result in the blended average of A and B
  – But $p(i)$ computed up front over all chunks