How does 6.1 differ from 6.0?

Dan Bothell
Carnegie Mellon University
db30@andrew.cmu.edu

Chunks do not have a type!

- A chunk is a set of slots with 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

Doesn’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

- Any isa provided is not used by the model!
  - NOT a test in a production condition
  - NOT a component of a request to a module

- Essentially a chunk-type is just a declaration not a constraint

Example chunk output

```
(chunk-type test slot1 slot2 slot3)
(define-chunks (chunk isa test slot1 "value"))
(pprint-chunks chunk)
```

In 6.0

```
CHUNK
  ISA TEST
   SLOT1  "value"
   SLOT2  NIL
   SLOT3  NIL
```

In 6.1

```
CHUNK
   SLOT1  "value"
```

Make chunk-types more useful in new role

- Now allows multiple inheritance
- Invalid slots for a specified type only lead to warnings in chunk and production definitions
- Implicit inclusion of default slot values from a chunk-type occurs in both chunk and production definitions now instead of just chunk definitions

Example model showing a default slot value being used

```
(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
   ==>
   !stop!
   !eval! (buffer-chunk goal))

 (pp)

 (run 1))
```

ACT-R 6.0

```
EXAMPLE
   ISA EXAMPLE
    SLOT  T

(P E1
    ?GOAL>
        BUFFER EMPTY
 ==>
    +GOAL>
        ISA EXAMPLE
)
(P E2
    =GOAL>
        ISA EXAMPLE
 ==>
    !STOP!
    !EVAL! (BUFFER-CHUNK GOAL)
)
0.000   CONFLICT-RESOLUTION
0.050   PRODUCTION-FIRED E1
0.050   CLEAR-BUFFER GOAL
0.050   SET-BUFFER-CHUNK GOAL
0.050   CONFLICT-RESOLUTION
0.100   PRODUCTION-FIRED E2
GOAL: EXAMPLE0-0
EXAMPLE0-0
   ISA EXAMPLE
    SLOT  T
```

ACT-R 6.1

```
EXAMPLE
   SLOT  T

(P E1
   ?GOAL>
       BUFFER EMPTY
 ==>
   +GOAL>
       SLOT T
)
(P E2
   =GOAL>
       SLOT T
 ==>
   !STOP!
   !EVAL! (BUFFER-CHUNK GOAL)
)
0.000  CONFLICT-RESOLUTION
0.050  PRODUCTION-FIRED E1
0.050  CLEAR-BUFFER GOAL
0.050  SET-BUFFER-CHUNK GOAL
0.050  CONFLICT-RESOLUTION
0.100  PRODUCTION-FIRED E2
GOAL: CHUNK0-0
CHUNK0-0
   SLOT  T
```

New production action indicator *

- Since isa is optional in production definitions the distinction between a request and a "modification request" can't hinge on the isa
  - These are equivalent in 6.1 unlike 6.0

```
+goal> slot value
+goal> isa something slot value
```

- ***** is now used for modification requests

`+goal> slot value` is now `*goal> slot value`

New production action indicator @

- Remove the special case for the = action to do a buffer overwrite
- **@** is now used for the buffer overwrite actions

  `=buffer>` ***`chunk`*** is now **`@buffer>`** ***`chunk`***

Now there are no special cases in production actions

- Given these definitions

```
(chunk-type x slot)
(define-chunks (value isa chunk) (c isa x slot value))
```

- These production actions all do the same thing

```
=goal> isa x slot value
=goal> slot value
=goal> c
```

- These also do the same as above (through the goal module)

```
*goal> isa x slot value
*goal> slot value
*goal> c
```

- These are also all the same (but not the same as above)

```
+goal> isa x slot value
+goal> slot value
+goal> c
```

Module requests

- Chunk-type information not provided
  - All details must be in the slots
- For the PM modules all of the chunk-types now have a slot named cmd which is used to indicate the action
  - The value is the same name as the chunk-type
- The chunk-types have a default value for that slot which matches the type name
- Therefore specifying the isa still works since the default slot value will be added to a production definition
- Either of the following will work in 6.1

```
+manual>
  isa press-key
  key “a”
```

```
+manual>
  cmd press-key
  key “a”
```

Other changes

- Remove the p/p* distinction
  - Both commands still exist and do the same thing
  - Using p is recommended now for all productions
- Simplify production condition syntax
  - One buffer test and/or one query per buffer
- Cannot modify chunks in DM now
  - Wasn't recommended before, but now it's strictly enforced

Will a 6.0 model work as-is in 6.1?

- Probably, unless it uses:
  - Modification requests
  - Buffer overwrites
  - Productions which are differentiated only by isa tests
- There is a system parameter called :backwards which can be set to true to handle those situations

- Out of 48 test models with ACT-R 6.0
  - 41 work the “same” as-is (functionally the same but some minor differences in model output/trace information)
  - 48 work if the :backwards system parameter set
- 25 of those models are from the tutorial units
  - 21 of the tutorial models work the same as-is

Typical issue to fix

- Production conditions or Lisp code which differentiate based only on the isa

```
 (p needs-the-isa-1        (p needs-the-isa-2
   =goal>                    =goal>
     isa task1                 isa task2
  ==>                       ==>
   ...)                     ...)
```

```
(sdp-fct (list (no-output (sdm isa number)) :base-level 3))
```

- Setting the :backwards switch will handle that without changing the model

Things that will require changes to model/code

- Lisp code which tests chunk types
  - Calls to chunk-chunk-type or chunk-spec-chunk-type will need to test something in a slot of the chunk instead
- Most module implementations will require some change
  - Requests usually tested the chunk-type info

Having “types” of chunks now a modeling choice

- Could give all chunks a slot to hold a type value essentially replacing the isa with a real slot
  - May not work well if a type hierarchy desired
  - Possibility for errors due to partial matching and spreading activation (may be good or bad depending on needs)
- Previously, sharing a type meant a common underlying structure which suggests differentiating based on the slots a chunk has not the value in a slot
  - Give each type a unique slot with a default value
  - If the value isn’t a chunk no spreading activation issues
  - Slots don’t get partial matched
- Other options also possible