

Unit8 Model code Description

The example models for this unit either have no experiment code, or are driven by code that has been described in previous units. The assignment task's experiment code does not use any new ACT-R commands or capabilities. Thus, instead of describing any of that code here, this text is going to explain how the `bst-learn-ppm` model avoids using a `!bind!` to do a calculation and instead uses a request to the `imaginal` module to do so.

The `imaginal` module has a second buffer called **imaginal-action** which can be used by the modeler to make requests that perform custom actions. Those actions are typically used to modify the chunk in the **imaginal** buffer, replace the chunk in the **imaginal** buffer with a new one, or clear the **imaginal** buffer and report an error, but may perform any other arbitrary calculation desired. Those requests can also take time during which the `imaginal` module will be considered as busy. Note however, the **imaginal-action** buffer is not intended to be used for holding a chunk. The **imaginal** buffer is the cognitive interface for the `imaginal` module and the **imaginal-action** buffer exists only for the purpose of allowing modelers to create new operations which can manipulate the **imaginal** buffer.

There are two types of requests which can be made to the **imaginal-action** buffer: `simple-action` and `generic-action`. This model uses a `simple-action` to create a new chunk for the **imaginal** buffer. The `generic-action` is more powerful in terms of what it can do, but also requires more care and programming from the modeler in handling the action. The `generic-action` request is beyond the scope of the tutorial, and users interested in using that capability should consult the reference manual for details.

Here is the production from the model which uses a `simple-action` request to the **imaginal-action** buffer:

```
(p encode-line-current
  =goal>
    isa      try-strategy
    state    attending
  =imaginal>
    isa encoding
    goal-loc =goal-loc
  =visual>
    isa      line
    width    =current-len
  ?visual>
    state    free
==>
  =imaginal>
    length   =current-len

  +imaginal-action>
    isa simple-action
    action compute-difference

  =goal>
    state    consider-next
  +visual>
    isa      move-attention
    screen-pos =goal-loc)
```

A simple-action request to the **imaginal-action** buffer requires specifying one slot, action, which must name a Lisp function. When a simple-action request is made the imaginal module performs the following actions: the imaginal module is marked as busy, if the imaginal module is currently signaling an error that is cleared, the named function is called with no parameters, and the **imaginal** buffer is cleared. After the current imaginal action time (default of 200ms and set with the `:imaginal-delay` parameter) has passed the imaginal module will be marked as free and one of two actions will occur depending on what the specified function returned. If it returned the name of a chunk then that chunk will be placed into the **imaginal** buffer. If it returns any other value the **imaginal** buffer will remain empty and the imaginal module's error flag will be set to t.

Here is the `compute-difference` function which is called as a result of the request in the `encode-line-current` production:

```
(defun compute-difference ()
  (let* ((chunk (buffer-read 'imaginal))
        (new-chunk (copy-chunk-fct chunk)))
    (mod-chunk-fct new-chunk
      (list 'difference
            (abs (- (chunk-slot-value-fct chunk 'length)
                    (chunk-slot-value-fct chunk 'goal-length)))))))
```

It creates a copy of the chunk which is currently in the imaginal buffer using the ACT-R `copy-chunk` command and then sets the difference slot of that new chunk to be the difference between the length of the current stick and the length of the goal stick just as the `!bind!` did in the previous version of the model. That new chunk is returned from the function and thus will be put into the **imaginal** buffer after 200ms have passed.

Here is the segment from the trace showing the actions related to the simple-action request when the `encode-line-current` production fires:

```
2.141  PROCEDURAL          PRODUCTION-SELECTED ENCODE-LINE-CURRENT
2.191  PROCEDURAL          PRODUCTION-FIRED ENCODE-LINE-CURRENT
...
2.191  PROCEDURAL          MODULE-REQUEST IMAGINAL-ACTION
2.191  PROCEDURAL          CLEAR-BUFFER IMAGINAL-ACTION
...
2.191  IMAGINAL            CLEAR-BUFFER IMAGINAL
...
2.391  IMAGINAL            SET-BUFFER-CHUNK IMAGINAL ENCODING0-0-0
```

Except for the additional clearing of the **imaginal-action** buffer, which should not hold a chunk anyway, it performs the same actions as the normal **imaginal** buffer requests do to create new chunks.

One important thing to note about a simple-action request is that it will always clear the **imaginal** buffer. That means that the chunk currently in the buffer will become an element of the model's declarative memory at that time. In this model that does not matter because it is not retrieving those chunks later. However, in models where later retrieval is important, having intermediate chunks added to memory like that could cause problems. In those cases, one would

probably want to use the generic-action request to extend the imaginal capabilities because it does not clear the buffer automatically.