Unit 1 Code Description

This document (and the corresponding documents in future units) will describe any ACT-R commands used in the unit's models which are not described in the main unit text, the code that implements the task/experiment for the models in that unit, how ACT-R is interfaced with the task/experiment, and describe additional commands that one can use interactively when working with ACT-R (most of the information available through the tools of the Environment can also be obtained directly from commands).

For this unit there are no tasks for the models and all of them run simply by using the ACT-R run command. They start with a predefined goal that is specifically placed into the goal buffer, they have a given set of declarative memories, and the result is a modification of the goal chunk representing the processing that took place. Later units will include tasks which have much more involved perception and action, but for this unit all of the models have the same basic structure as shown here in a solution to the multi-column addition model one was to write for unit1:

```
(clear-all)

(define-model tutor-model

(sgp :esc t :lf .05 :trace-detail medium)

;; Add Chunk-types here

(chunk-type addition-fact addend1 addend2 sum)
(chunk-type add-pair one1 ten1 one2 ten2 ten-ans one-ans carry)

;; Add Chunks here

(add-dm
 (fact17 isa addition-fact addend1 1 addend2 7 sum 8)
 (fact34 isa addition-fact addend1 3 addend2 4 sum 7)
 (fact67 isa addition-fact addend1 6 addend2 7 sum 13)
 (fact103 isa addition-fact addend1 10 addend2 3 sum 13)
 (goal isa add-pair ten1 3 one1 6 ten2 4 one2 7))

;; Add productions here

(p start-pair
  =goal>
    ISA add-pair
    one1 =num1
    one2 =num2
    one-ans nil
==>
  =goal>
    one-ans busy
  +retrieval>
```

```
    ISA addition-fact
    addend1 =num1
    addend2 =num2)

(p add-ones
  =goal>
    ISA add-pair
    one-ans busy
    one1 =num1
    one2 =num2
  =retrieval>
    ISA addition-fact
    addend1 =num1
    addend2 =num2
    sum =sum
==>
  =goal>
    one-ans =sum
    carry busy
  +retrieval>
    ISA addition-fact
    addend1 10
    sum =sum)

(p process-carry
  =goal>
    ISA add-pair
    ten1 =num1
    ten2 =num2
    carry busy
    one-ans =ones
  =retrieval>
    ISA addition-fact
    addend1 10
    sum =ones
    addend2 =remainder
==>
  =goal>
    carry 1
    ten-ans busy
    one-ans =remainder
  +retrieval>
    ISA addition-fact
    addend1 =num1
    addend2 =num2)

(p no-carry
  =goal>
    ISA add-pair
    ten1 =num1
    ten2 =num2
    one-ans =ones
```

```
    carry busy
  ?retrieval>
    buffer failure
==>
  =goal>
    carry nil
    ten-ans busy
  +retrieval>
    ISA addition-fact
    addend1 =num1
    addend2 =num2)

(p add-tens-done
  =goal>
    ISA add-pair
    ten-ans busy
    carry nil
  =retrieval>
    ISA addition-fact
    sum =sum
==>
  =goal>
    ten-ans =sum)

(p add-tens-carry
  =goal>
    ISA add-pair
    carry 1
    ten-ans busy
  =retrieval>
    ISA addition-fact
    sum =sum
==>
  =goal>
    carry nil
  +retrieval>
    ISA addition-fact
    addend1 1
    addend2 =sum)

(goal-focus goal)
)
```

It starts with a call to a function called **clear-all** followed by a call to the command **define-model**. Inside the call to **define-model** there is a call to a command called **sgp**, then all of the components of the model are defined (chunk-types, chunks, and productions) and finally the starting goal chunk gets set using the **goal-focus** command.

Most of these commands were described in the main unit text, which also described some tools in the ACT-R Environment for inspecting and debugging models. Here we will provide some

additional details for some of those commands, describe the sgp command, and show how one can also perform some of the actions that were done using the Environment tools from the prompt in the ACT-R window. In later units, we will also show how those commands could be accessed remotely using the included Python module as an example.

Additional ACT-R command details

Clear-all

The **clear-all** command is used to set the ACT-R software to its initial state. It removes all of the models that are currently defined, returns the clock to time 0, and removes any events which are on the event queue. If a model (or set of models) is contained within a single file, then one probably should call **clear-all** in that file to make sure ACT-R is initialized before defining the model components.

An additional side effect of the **clear-all** command is that it records the name of the file that contains it when it is loaded. That is how the **reload** command knows which file to "reload".

Define-model

The **define-model** command takes one required parameter which is the name of a new model to create and then an arbitrary number of other parameters which are the commands which create the initial conditions for the model. The name should be a symbol and must be unique with respect to other models that are currently defined. When a **reset** happens, all of the commands specified inside of the **define-model** are reevaluated for that model in the order they were specified. Each call to **define-model** creates a new model which is independent of the other models, but all of the models will run in parallel when the **run** command is called.

SGP

The **sgp** command is used to set or show the model specific parameters (it stands for set/show general parameters). The parameters for a model control many different things. Some are used in equations that control the performance of the model's cognitive modules, others are there to help the modeler with debugging by changing the outputting of information or the seed of the pseudorandom number generator, and others are available to provide ways that the modeler can extend or modify internal ACT-R mechanisms. The details of all of the parameters can be found in the reference manual.

When using **sgp** to set parameters the syntax is to specify a parameter and then the new value you want to assign to that parameter. Any number of parameters and values may be specified in a single call to **sgp**. All of the parameters in ACT-R begin with a ":" (in Lisp syntax they are keywords). All of the unit1 models have a call to **sgp** similar to this:

```
(sgp :esc t :lf .05 :trace-detail medium)
```

That is setting three parameters**: :esc**, **:lf**, and **:trace-detail**. The first two together are specifying that retrieval requests will always take 50ms to complete, but further details on those are beyond the scope of this unit and will be discussed fully in later units. The third parameter being set, **:trace-detail**, controls how much information is shown in the trace when a model runs. The default value is **medium**, and that is also how it is being set in the example above. The other values that it can have are **high** and **low**. When it is set to **high**, effectively every action the model does shows in the trace. Here is the trace of the two-digit addition model with **:trace-detail** set to **high**:

```
0.000   GOAL                    SET-BUFFER-CHUNK GOAL GOAL REQUESTED NIL
0.000   PROCEDURAL              CONFLICT-RESOLUTION
0.000   PROCEDURAL              PRODUCTION-SELECTED START-PAIR
0.000   PROCEDURAL              BUFFER-READ-ACTION GOAL
0.050   PROCEDURAL              PRODUCTION-FIRED START-PAIR
0.050   PROCEDURAL              MOD-BUFFER-CHUNK GOAL
0.050   PROCEDURAL              MODULE-REQUEST RETRIEVAL
0.050   PROCEDURAL              CLEAR-BUFFER RETRIEVAL
0.050   DECLARATIVE             START-RETRIEVAL
0.050   PROCEDURAL              CONFLICT-RESOLUTION
0.100   DECLARATIVE             RETRIEVED-CHUNK FACT67
0.100   DECLARATIVE             SET-BUFFER-CHUNK RETRIEVAL FACT67
0.100   PROCEDURAL              CONFLICT-RESOLUTION
0.100   PROCEDURAL              PRODUCTION-SELECTED ADD-ONES
0.100   PROCEDURAL              BUFFER-READ-ACTION GOAL
0.100   PROCEDURAL              BUFFER-READ-ACTION RETRIEVAL
0.150   PROCEDURAL              PRODUCTION-FIRED ADD-ONES
0.150   PROCEDURAL              MOD-BUFFER-CHUNK GOAL
0.150   PROCEDURAL              MODULE-REQUEST RETRIEVAL
0.150   PROCEDURAL              CLEAR-BUFFER RETRIEVAL
0.150   DECLARATIVE             START-RETRIEVAL
0.150   PROCEDURAL              CONFLICT-RESOLUTION
0.200   DECLARATIVE             RETRIEVED-CHUNK FACT103
0.200   DECLARATIVE             SET-BUFFER-CHUNK RETRIEVAL FACT103
0.200   PROCEDURAL              CONFLICT-RESOLUTION
0.200   PROCEDURAL              PRODUCTION-SELECTED PROCESS-CARRY
0.200   PROCEDURAL              BUFFER-READ-ACTION GOAL
0.200   PROCEDURAL              BUFFER-READ-ACTION RETRIEVAL
0.250   PROCEDURAL              PRODUCTION-FIRED PROCESS-CARRY
0.250   PROCEDURAL              MOD-BUFFER-CHUNK GOAL
0.250   PROCEDURAL              MODULE-REQUEST RETRIEVAL
0.250   PROCEDURAL              CLEAR-BUFFER RETRIEVAL
0.250   DECLARATIVE             START-RETRIEVAL
0.250   PROCEDURAL              CONFLICT-RESOLUTION
0.300   DECLARATIVE             RETRIEVED-CHUNK FACT34
0.300   DECLARATIVE             SET-BUFFER-CHUNK RETRIEVAL FACT34
0.300   PROCEDURAL              CONFLICT-RESOLUTION
0.300   PROCEDURAL              PRODUCTION-SELECTED ADD-TENS-CARRY
0.300   PROCEDURAL              BUFFER-READ-ACTION GOAL
0.300   PROCEDURAL              BUFFER-READ-ACTION RETRIEVAL
0.350   PROCEDURAL              PRODUCTION-FIRED ADD-TENS-CARRY
```

```
0.350   PROCEDURAL              MOD-BUFFER-CHUNK GOAL
0.350   PROCEDURAL              MODULE-REQUEST RETRIEVAL
0.350   PROCEDURAL              CLEAR-BUFFER RETRIEVAL
0.350   DECLARATIVE             START-RETRIEVAL
0.350   PROCEDURAL              CONFLICT-RESOLUTION
0.400   DECLARATIVE             RETRIEVED-CHUNK FACT17
0.400   DECLARATIVE             SET-BUFFER-CHUNK RETRIEVAL FACT17
0.400   PROCEDURAL              CONFLICT-RESOLUTION
0.400   PROCEDURAL              PRODUCTION-SELECTED ADD-TENS-DONE
0.400   PROCEDURAL              BUFFER-READ-ACTION GOAL
0.400   PROCEDURAL              BUFFER-READ-ACTION RETRIEVAL
0.450   PROCEDURAL              PRODUCTION-FIRED ADD-TENS-DONE
0.450   PROCEDURAL              MOD-BUFFER-CHUNK GOAL
0.450   PROCEDURAL              CLEAR-BUFFER RETRIEVAL
0.450   PROCEDURAL              CONFLICT-RESOLUTION
0.450   ------                  Stopped because no events left to process
```

That can be very useful when debugging a model but it can be a bit too much at other times. Here is the same model running with a **medium** level of **:trace-detail** (which is the default value):

```
0.000   GOAL                    SET-BUFFER-CHUNK GOAL GOAL REQUESTED NIL
0.000   PROCEDURAL              CONFLICT-RESOLUTION
0.050   PROCEDURAL              PRODUCTION-FIRED START-PAIR
0.050   PROCEDURAL              CLEAR-BUFFER RETRIEVAL
0.050   DECLARATIVE             START-RETRIEVAL
0.050   PROCEDURAL              CONFLICT-RESOLUTION
0.100   DECLARATIVE             RETRIEVED-CHUNK FACT67
0.100   DECLARATIVE             SET-BUFFER-CHUNK RETRIEVAL FACT67
0.100   PROCEDURAL              CONFLICT-RESOLUTION
0.150   PROCEDURAL              PRODUCTION-FIRED ADD-ONES
0.150   PROCEDURAL              CLEAR-BUFFER RETRIEVAL
0.150   DECLARATIVE             START-RETRIEVAL
0.150   PROCEDURAL              CONFLICT-RESOLUTION
0.200   DECLARATIVE             RETRIEVED-CHUNK FACT103
0.200   DECLARATIVE             SET-BUFFER-CHUNK RETRIEVAL FACT103
0.200   PROCEDURAL              CONFLICT-RESOLUTION
0.250   PROCEDURAL              PRODUCTION-FIRED PROCESS-CARRY
0.250   PROCEDURAL              CLEAR-BUFFER RETRIEVAL
0.250   DECLARATIVE             START-RETRIEVAL
0.250   PROCEDURAL              CONFLICT-RESOLUTION
0.300   DECLARATIVE             RETRIEVED-CHUNK FACT34
0.300   DECLARATIVE             SET-BUFFER-CHUNK RETRIEVAL FACT34
0.300   PROCEDURAL              CONFLICT-RESOLUTION
0.350   PROCEDURAL              PRODUCTION-FIRED ADD-TENS-CARRY
0.350   PROCEDURAL              CLEAR-BUFFER RETRIEVAL
0.350   DECLARATIVE             START-RETRIEVAL
0.350   PROCEDURAL              CONFLICT-RESOLUTION
0.400   DECLARATIVE             RETRIEVED-CHUNK FACT17
0.400   DECLARATIVE             SET-BUFFER-CHUNK RETRIEVAL FACT17
```

```
0.400   PROCEDURAL            CONFLICT-RESOLUTION
0.450   PROCEDURAL            PRODUCTION-FIRED ADD-TENS-DONE
0.450   PROCEDURAL            CLEAR-BUFFER RETRIEVAL
0.450   PROCEDURAL            CONFLICT-RESOLUTION
0.450   ------                Stopped because no events left to process
```

In that trace we no longer see the individual condition tests and only some of the actions of the productions. Now, here is the same model run with a **low** setting for **:trace-detail**:

```
0.000   GOAL                  SET-BUFFER-CHUNK GOAL GOAL REQUESTED NIL
0.050   PROCEDURAL            PRODUCTION-FIRED START-PAIR
0.100   DECLARATIVE           SET-BUFFER-CHUNK RETRIEVAL FACT67
0.150   PROCEDURAL            PRODUCTION-FIRED ADD-ONES
0.200   DECLARATIVE           SET-BUFFER-CHUNK RETRIEVAL FACT103
0.250   PROCEDURAL            PRODUCTION-FIRED PROCESS-CARRY
0.300   DECLARATIVE           SET-BUFFER-CHUNK RETRIEVAL FACT34
0.350   PROCEDURAL            PRODUCTION-FIRED ADD-TENS-CARRY
0.400   DECLARATIVE           SET-BUFFER-CHUNK RETRIEVAL FACT17
0.450   PROCEDURAL            PRODUCTION-FIRED ADD-TENS-DONE
0.450   ------                Stopped because no events left to process
```

At **low** we only see production firings and buffer settings. The setting of **:trace-detail** does not change how the model actually runs. It only affects how the trace is displayed to the modeler and which events are available to "step" to using the Stepper tool in the ACT-R Environment.

If one wants to completely turn off the model trace, there is another parameter which can be set to do so, and that will be described in a future unit.

To get a parameter's current value using **sgp** only the name of the parameter (or parameters) should be specified. When all of the values passed to **sgp** are the names of parameters it will print out the details of those parameters and return a list of their current values. Typically that is not necessary when defining a model (with one exception which can be very helpful for debugging a model and that will be described in a later unit), but it can be used at the ACT-R prompt to inspect the model settings (there is also an inspector in the Environment for doing so). Here is an example which is checking the values of the **:trace-detail** and **:lf** parameters:

```
? (sgp :trace-detail :lf)
:TRACE-DETAIL MEDIUM (default MEDIUM) : Determines which events show in the trace
:LF 0.05 (default 1.0) : Latency Factor
(MEDIUM 0.05)
```

If no parameters are provided to **sgp** then it will print out all of the parameters and their details.

Interacting with a model

In this unit we used the ACT-R Environment tools to load, run, reset, and inspect the model components. Most of the GUI tools in the Environment are using commands which are also available to the modeler for interacting with the model from the command prompt or in experiment or task code. Here we will describe some of the commands that correspond to the Environment tools used for this unit with respect to using them from the ACT-R command prompt (as was shown above for **sgp**). In future units we will discuss how those commands can also be used in experiments or tasks written in Lisp or Python because those are the languages for which an interface to the commands is included with the software (other languages could also be supported but the details needed to do so are beyond the scope of this tutorial).

Loading a model

To load the models in this unit we used the "Load ACT-R code" button to pick a file to load. There is a command in ACT-R which can also be used to load a model file which is called **load-act-r-model**. It requires one parameter which is a string specifying the pathname of the file to load. Typically, the full pathname of the file must be specified, but it does accept a simplified specification for files located in the directory containing the ACT-R files which is based on the Lisp logical pathname convention. The format for those relative pathnames is to start with ACT-R: and then follow that with the file name or subdirectories separated by semicolons and then the file name. Here is an example of loading the count.lisp model from the default ACT-R distribution:

```
? (load-act-r-model "ACT-R:tutorial;unit1;count.lisp")
T
```

The **T** printed after the call is the return value from the command, and **T** is the Lisp symbol for true, which for this command means it was successful in loading the file. If the file is not found or there is an error in loading it then it will print a warning indicating the issue and the return value would be **nil** instead.

Resetting and Reloading

Instead of pressing the buttons to reset or reload a model one can call the corresponding commands, which are named reset and reload. They require no parameters and will return **T** if successful:

```
? (reset)
T
? (reload)
T
```

Running the model

There are actually multiple commands which can be used to run a model based on how it should determine when to stop running. The one that corresponds to the button on the Control Panel is called **run**. It requires one parameter which is the maximum number of seconds to run the model, but it will stop earlier if the model has no more actions to perform. In later units we will introduce more of the running commands and show them being used in creating tasks that automatically run the model as needed. Here is an example of using the **run** command at the prompt after loading the semantic model:

```
? (run 1)
     0.000   GOAL                 SET-BUFFER-CHUNK GOAL G1 NIL
     0.000   PROCEDURAL           CONFLICT-RESOLUTION
     0.050   PROCEDURAL           PRODUCTION-FIRED INITIAL-RETRIEVE
     0.050   PROCEDURAL           CLEAR-BUFFER RETRIEVAL
     0.050   DECLARATIVE          start-retrieval
     0.050   PROCEDURAL           CONFLICT-RESOLUTION
     0.100   DECLARATIVE          RETRIEVED-CHUNK P14
     0.100   DECLARATIVE          SET-BUFFER-CHUNK RETRIEVAL P14
     0.100   PROCEDURAL           CONFLICT-RESOLUTION
     0.150   PROCEDURAL           PRODUCTION-FIRED DIRECT-VERIFY
     0.150   PROCEDURAL           CLEAR-BUFFER RETRIEVAL
     0.150   PROCEDURAL           CONFLICT-RESOLUTION
     0.150   ------               Stopped because no events left to process
0.15
24
NIL
```

After the trace there are multiple return values shown from the run command which indicate how long it ran, how many events occurred during the run, and whether it ended unexpectedly or not (a value of **nil** means a successful ending because it was not unexpected due to a break action or other problem).

Inspecting model components

During the unit we inspected the contents of the buffers, the model's declarative memory, and checked why productions did not match. All of those can also be done using commands.

Buffers

You can use the command named **buffer-chunk** to find the names of the chunks in the buffers and inspect their contents. Calling it without any parameters will show all of the buffers and the chunks they contain returning a list of the buffer name and buffer chunk lists:

```
? (buffer-chunk)
GOAL: G1-0
TEMPORAL: NIL
AURAL: NIL
AURAL-LOCATION: NIL
MANUAL: NIL
VOCAL: NIL
RETRIEVAL: NIL
PRODUCTION: NIL
```

```
IMAGINAL: NIL
VISUAL: NIL
VISUAL-LOCATION: NIL
IMAGINAL-ACTION: NIL
((GOAL G1-0) (TEMPORAL) (AURAL) (AURAL-LOCATION) (MANUAL) (VOCAL) (RETRIEVAL)
(PRODUCTION) (IMAGINAL) (VISUAL) (VISUAL-LOCATION) (IMAGINAL-ACTION))
```

If you call it with the name of a buffer (or multiple buffers), then it will print out the chunks in the named buffers and return the list of the names of those chunks:

```
? (buffer-chunk goal)
GOAL: G1-0
G1-0
   OBJECT  CANARY
   CATEGORY  BIRD
   JUDGMENT  YES

(G1-0)
```

Declarative Memory

You can also inspect declarative memory from the command prompt. The command **dm** will print out all of the chunks in the model's declarative memory and return a list with the names of those chunks. You can also specify the names of chunks as parameters to the **dm** command and only those chunks will be printed. Here are some examples using the count model:

```
? (dm)
FIRST-GOAL
   START  2
   END  4

F
   FIRST  5
   SECOND  6

E
   FIRST  4
   SECOND  5

D
   FIRST  3
   SECOND  4

C
   FIRST  2
   SECOND  3

B
   FIRST  1
   SECOND  2

(FIRST-GOAL F E D C B)

? (dm b e f)
B
```

```
   FIRST  1
   SECOND  2

E
   FIRST  4
   SECOND  5

F
   FIRST  5
   SECOND  6

(B E F)
?
```

It is also possible to search declarative memory using the command **sdm**. Its parameters are a chunk specification just as one would specify in a retrieval request in a production, but without the +retrieval> indicator at the beginning. It prints out only those chunks from the model's declarative memory which match that specification and returns the list of their names. Here are some examples using the semantic model:

```
? (sdm object shark)
P1
   OBJECT  SHARK
   VALUE  TRUE
   ATTRIBUTE  DANGEROUS

P2
   OBJECT  SHARK
   VALUE  SWIMMING
   ATTRIBUTE  LOCOMOTION

P3
   OBJECT  SHARK
   VALUE  FISH
   ATTRIBUTE  CATEGORY

(P1 P2 P3)

? (sdm - attribute category - attribute nil)
P1
   OBJECT  SHARK
   VALUE  TRUE
   ATTRIBUTE  DANGEROUS

P2
   OBJECT  SHARK
   VALUE  SWIMMING
   ATTRIBUTE  LOCOMOTION

P4
   OBJECT  SALMON
   VALUE  TRUE
   ATTRIBUTE  EDIBLE

P5
   OBJECT  SALMON
```

```
   VALUE  SWIMMING
   ATTRIBUTE  LOCOMOTION

P7
   OBJECT  FISH
   VALUE  GILLS
   ATTRIBUTE  BREATHE

P8
   OBJECT  FISH
   VALUE  SWIMMING
   ATTRIBUTE  LOCOMOTION

P10
   OBJECT  ANIMAL
   VALUE  TRUE
   ATTRIBUTE  MOVES

P11
   OBJECT  ANIMAL
   VALUE  TRUE
   ATTRIBUTE  SKIN

P12
   OBJECT  CANARY
   VALUE  YELLOW
   ATTRIBUTE  COLOR

P13
   OBJECT  CANARY
   VALUE  TRUE
   ATTRIBUTE  SINGS

P15
   OBJECT  OSTRICH
   VALUE  FALSE
   ATTRIBUTE  FLIES

P16
   OBJECT  OSTRICH
   VALUE  TALL
   ATTRIBUTE  HEIGHT

P18
   OBJECT  BIRD
   VALUE  TRUE
   ATTRIBUTE  WINGS

P19
   OBJECT  BIRD
   VALUE  FLYING
   ATTRIBUTE  LOCOMOTION

(P1 P2 P4 P5 P7 P8 P10 P11 P12 P13 P15 P16 P18 P19)
```

One thing to note about the second example is that it specifies both that the attribute slot is not category and also not nil (which means the chunk must have some value for the slot). Here is what it returns if it were to only specify that the attribute slot is not category:

```
? (sdm - attribute category)
...
(G1 G2 G3 P1 P2 P4 P5 P7 P8 P10 P11 P12 P13 P15 P16 P18 P19 SHARK DANGEROUS
LOCOMOTION SWIMMING FISH SALMON EDIBLE BREATHE GILLS ANIMAL MOVES SKIN CANARY
COLOR SINGS BIRD OSTRICH FLIES HEIGHT TALL WINGS FLYING TRUE FALSE)
```

In this case it also finds all of the chunks which do not have an attribute slot because chunks without the slot fail the "attribute category" test and thus the negation of that is then true. That is something to be careful about when using the negation modifier in specifying retrieval requests in your productions as well.

Testing why not? for productions

The a command to test whether a production matches the current state is called **whynot**. If you pass it no parameters it will print out each chunk in the model's procedural memory along with either an indication that it matches or a reason why it does not match. You can also provide it with specific production names to test. It returns a list of the names of the productions which do match the current state. Here is an example using productions from the semantic model after it has been reset:

```
? (whynot direct-verify fail)

Production DIRECT-VERIFY does NOT match.
(P DIRECT-VERIFY
   =GOAL>
       OBJECT =OBJ
       CATEGORY =CAT
       JUDGMENT PENDING
   =RETRIEVAL>
       OBJECT =OBJ
       ATTRIBUTE CATEGORY
       VALUE =CAT
 ==>
   =GOAL>
       JUDGMENT YES
)
It fails because:
The GOAL buffer is empty.

Production FAIL does NOT match.
(P FAIL
   =GOAL>
       OBJECT =OBJ1
       CATEGORY =CAT
       JUDGMENT PENDING
   ?RETRIEVAL>
       BUFFER FAILURE
 ==>
```

```
   =GOAL>
       JUDGMENT NO
)
It fails because:
The GOAL buffer is empty.
NIL
```

Other model commands

Any of the commands that are specified in the model definition can also be called from the ACT-R command prompt. The **add-dm** and **p** commands are not usually called from the prompt since you want that knowledge to be included in the model when it starts running, but a command like **goal-focus** can occasionally be useful if you want to change the goal chunk for a model and run it again. That could have been convenient for the semantic model because instead of changing the file and reloading it to switch the goals one could have run the different goals sequentially by just calling goal-focus to change the chunk in the goal buffer and then run it again:

```
? (run 1)
     0.000   GOAL                 SET-BUFFER-CHUNK GOAL G1 NIL
     0.000   PROCEDURAL           CONFLICT-RESOLUTION
     0.050   PROCEDURAL           PRODUCTION-FIRED INITIAL-RETRIEVE
     0.050   PROCEDURAL           CLEAR-BUFFER RETRIEVAL
     0.050   DECLARATIVE          start-retrieval
     0.050   PROCEDURAL           CONFLICT-RESOLUTION
     0.100   DECLARATIVE          RETRIEVED-CHUNK P14
     0.100   DECLARATIVE          SET-BUFFER-CHUNK RETRIEVAL P14
     0.100   PROCEDURAL           CONFLICT-RESOLUTION
     0.150   PROCEDURAL           PRODUCTION-FIRED DIRECT-VERIFY
     0.150   PROCEDURAL           CLEAR-BUFFER RETRIEVAL
     0.150   PROCEDURAL           CONFLICT-RESOLUTION
     0.150   ------               Stopped because no events left to process
0.15
24
NIL
? (goal-focus g2)
G2
? (run 1)
     0.150   GOAL                 SET-BUFFER-CHUNK GOAL G2 NIL
     0.150   PROCEDURAL           CONFLICT-RESOLUTION
     0.200   PROCEDURAL           PRODUCTION-FIRED INITIAL-RETRIEVE
     0.200   PROCEDURAL           CLEAR-BUFFER RETRIEVAL
     0.200   DECLARATIVE          start-retrieval
     0.200   PROCEDURAL           CONFLICT-RESOLUTION
     0.250   DECLARATIVE          RETRIEVED-CHUNK P14
     0.250   DECLARATIVE          SET-BUFFER-CHUNK RETRIEVAL P14
     0.250   PROCEDURAL           CONFLICT-RESOLUTION
     0.300   PROCEDURAL           PRODUCTION-FIRED CHAIN-CATEGORY
     0.300   PROCEDURAL           CLEAR-BUFFER RETRIEVAL
     0.300   DECLARATIVE          start-retrieval
     0.300   PROCEDURAL           CONFLICT-RESOLUTION
     0.350   DECLARATIVE          RETRIEVED-CHUNK P20
     0.350   DECLARATIVE          SET-BUFFER-CHUNK RETRIEVAL P20
     0.350   PROCEDURAL           CONFLICT-RESOLUTION
```

```
     0.400   PROCEDURAL              PRODUCTION-FIRED DIRECT-VERIFY
     0.400   PROCEDURAL              CLEAR-BUFFER RETRIEVAL
     0.400   PROCEDURAL              CONFLICT-RESOLUTION
     0.400   ------                  Stopped because no events left to process
0.25
36
NIL
? (goal-focus g3)
G3
? (run 1)
     0.400   GOAL                    SET-BUFFER-CHUNK GOAL G3 NIL
     0.400   PROCEDURAL              CONFLICT-RESOLUTION
     0.450   PROCEDURAL              PRODUCTION-FIRED INITIAL-RETRIEVE
     0.450   PROCEDURAL              CLEAR-BUFFER RETRIEVAL
     0.450   DECLARATIVE             start-retrieval
     0.450   PROCEDURAL              CONFLICT-RESOLUTION
     0.500   DECLARATIVE             RETRIEVED-CHUNK P14
     0.500   DECLARATIVE             SET-BUFFER-CHUNK RETRIEVAL P14
     0.500   PROCEDURAL              CONFLICT-RESOLUTION
     0.550   PROCEDURAL              PRODUCTION-FIRED CHAIN-CATEGORY
     0.550   PROCEDURAL              CLEAR-BUFFER RETRIEVAL
     0.550   DECLARATIVE             start-retrieval
     0.550   PROCEDURAL              CONFLICT-RESOLUTION
     0.600   DECLARATIVE             RETRIEVED-CHUNK P20
     0.600   DECLARATIVE             SET-BUFFER-CHUNK RETRIEVAL P20
     0.600   PROCEDURAL              CONFLICT-RESOLUTION
     0.650   PROCEDURAL              PRODUCTION-FIRED CHAIN-CATEGORY
     0.650   PROCEDURAL              CLEAR-BUFFER RETRIEVAL
     0.650   DECLARATIVE             start-retrieval
     0.650   PROCEDURAL              CONFLICT-RESOLUTION
     0.700   DECLARATIVE             RETRIEVAL-FAILURE
     0.700   PROCEDURAL              CONFLICT-RESOLUTION
     0.750   PROCEDURAL              PRODUCTION-FIRED FAIL
     0.750   PROCEDURAL              CLEAR-BUFFER RETRIEVAL
     0.750   PROCEDURAL              CONFLICT-RESOLUTION
     0.750   ------                  Stopped because no events left to process
0.35
48
NIL
```