Declarative Memory

Models and Modules

Mon, 19/05/2008 - 20:16 — terry

Declarative Memory: This is the standard system for storing and retrieving 'chunks' of information in ACT-R. This is generally used to model information that is explicitly represented.

The following example shows information being put into declarative memory and then being retrieved.

```
from ccm.lib.actr import *            # allows use of Python ACT-R

class SimpleModel2(ACTR):             # everything in here defines the model

  # first, define the modules within the model
  goal=Buffer()                    # goal buffer
  retrieval=Buffer()               # a buffer for storing retrieved chunks
  memory=Memory(retrieval)         # the declarative memory system

  # now define the productions
  def remember(goal='remember'):
    memory.request('fact')         # attempt to recall a matching chunk
    goal.set('remembering')

  def remembered(goal='remembering',retrieval='fact ?a ?b ?c'):
      print a,b,c
      goal.set('done')

# now we actually create an instance of the model
model=SimpleModel2()

# add information into declarative memory
model.memory.add('fact cats are cute')
model.memory.add('fact people are strange')
model.memory.add('fact cars go fast')

model.goal.set('remember')       # and initialize its goal buffer
model.run()                      # and run it
```

For details on how to specify chunks and working with the matching rules (i.e. specifying what sort of memory chunks to retrieve), see the previous section.

By default, the declarative memory system randomly choses one of the chunks which match the request. For more complex models, there are a variety of sub-modules which can adjust the activation of the chunks in memory, giving them different probabilities of being retrieved.

The first step is always to create a retrieval buffer and a basic declarative memory system to go with it. The memory system has a variety of parameters:

- latency: controls the relationship between activation and how long it taks to recall the chunk
- threshold: chunks must have activation greater than or equal to this to be recalled (can be set to None)
- maximum_time: no retrieval will take longer than this amount of time
- finst_size: the FINST system allows you to recall items that have not been recently recalled. This controls how many recent items are remembered
- finst_time: this controls how recent recalls must be for them to be included in the FINST system.

```
retrieval=Buffer()
memory=Memory(retrieval,latency=0.05,threshold=0,maximum_time=10.0,finst_size=0,finst_time=3.0)
```

If no sub-modules are used, then the activation of all chunks is always zero. The following systems adjust the activation in various ways. They are shown along with the default values for their parameters.

```
# standard random noise
dm_n=DMNoise(memory,noise=0.3,baseNoise=0.0)

# the standard base level learning system
#  if limit is set to a number, then the hybrid optimized equation
#  from (Petrov, 2006) is used.
dm_bl=DMBaseLevel(memory,decay=0.5,limit=None)

# the spacing effect system from (Pavlik and Anderson, 2005)
dm_space=DMSpacing(memory,decayScale=0.0,decayIntercept=0.5)

# the standard fan-effect spreading activation system
dm_spread=DMSpreading(memory,goal)  # specify the buffer(s) to spread from
   # other parameters are configured like this:
   #   dm_spread.strength=1
   #   dm_spread.weight[goal]=0.3
```

Comments

You do not have permission to add comments.