Cognitive Modeling (2009-2010)

School of Informatics, University of Edinburgh
Lecturer: Sharon Goldwater

Assignment 0

Due date:	25 Jan
Weighting:	0% of total mark

This assignment will not count toward your final mark for the course, it is just intended to help you get familiar with Cogent, as introduced in Lecture 3 and in Cooper (2002, Ch. 2). You will work with a simple model of memory using Cogent. The due date is listed as 25th January because I will set aside some time during class on the 26th in case there are questions or problems that come up, and also so that you will make sure to identify any problems you're having well before the deadline for Assignment 1. Since this assignment does not count towards your final mark, you are free to collaborate as much as you want.

1 Getting Started with Cogent

Log on to your Dice account. Type `cogent` on the command prompt. If this is the first time you're launching Cogent, it will take you through an installation procedure. In the first dialog box that pops up, select `standard installation`. In the next box select the following paths:

User files and directories:	[your choice]
System files and folders:	[leave unchanged]
OOS executable:	[leave unchanged]

2 Building the Modal Model of Memory

For the questions in this assignment, you will be working with a pre-built version of the Modal Model discussed in lecture. However, if you want to gain some familiarity with Cogent's basic functionality and user interface before proceeding to the questions, a good way to do so is to build some parts of the model yourself by working through (at least some of) Exercises 2.1–2.14 in Cooper (2002, Ch. 2). These exercises introduce core aspects of the Modal Model of memory, the topic of the rest of this assignment. There are a few inconsistencies between the instructions in the text and the current version of Cogent; see the Errata sheet I've posted on the course web site for corrections to the text.

3 Exploring the Modal Model of Memory

Download the following file, which contains the model that you will work with in this assignment:
`http://www.inf.ed.ac.uk/teaching/courses/cm/assignments/cm_asgn0.tar.gz`

Use `gunzip` and `tar` to unpack this file in the `projects` subdirectory of your Cogent directory (this is the user directory you specified during the installation). Upon restarting Cogent, you should see a research program called `Assignment 0` in the Cogent root window. Select this program and doubleclick on the only model within this research program, called `Modal Model`. This model corresponds to the final version of the model developed in Cooper (2002, Ch. 2). The model consists of two parts: `Task Environment` models the experimental setting, `Subject Model` models the memory system of the subject.

Open the two memory components of the `Subject Model`, i.e., the short term store `STS` and the long term store `LTS` (model elements can be opened by doubleclicking on them). Now select the `Current Contents` tab of both stores. Use the `Step` button (the icon marked '>') to run the model, one cycle at a time. At each step observe how the contents of the two stores changes.

> **Question 1**
> What is the maximum number of words that the `STS` buffer can store? Why does this limit make sense? How is it implemented in Cogent?

> **Question 2**
> Does the `LTS` buffer also have a limited capacity? If it doesn't, then why don't all words in `STS` simply end up in `LTS`? What is the role of the `Rehearsal` process in achieving this behavior?

Now run the model to completion by pressing the icon marked '≫'. Inspect the output by selecting the `Output Positions` component in the `Task Environment` part of the model. Click on the `Results` tab to see what the model has recalled: for instance, `data(recall,18,100)` means that the model has recalled the word on position 18 with 100% accuracy.

> **Question 3**
> Select the `Current Graph` tab of the `Output Positions` element. Why does the graph look the way it does? How does it change if you run the model 10 times?

> **Question 4**
> If you run the model multiple times (as in the last question), you will realize that its behavior varies randomly from one run to the other, i.e., it doesn't always recall the same words. Explain which parts of the model are responsible for the randomness.

4 Parameterizing the Modal Model

You shouldn't modify the original version of the model. Instead go to the research program manager and create a copy of the modal model. Work on this copy for the questions in this section.

As explained in Cooper (2002, pp. 73–75), the performance of the model is currently too good to be realistic: the `LTS` buffer recalls items perfectly (i.e., it never forgets anything). Doubleclick on the `LTS` element and select the `Properties` tab. One of the properties is `decay` which specifies how items decay from the buffer. Decay is currently turned off.

> **Question 5**
> Set `decay` to `half-life` and the `decay constant` to 30. Reset the model by pressing the '`o`' button. Now run the model to completion 20 times. (You can do this automatically by clicking on `OOS window...` from the `View` menu. On the `Current Script` tab, enter `20` in the text box, then save and close the OOS window. Now when you click the '≫' button, 20 trials will be run instead of only one.) How does the recall graph look now? Explain why. Repeat this for the other two types of `decay`, i.e., `linear` and `fixed`. Does this change the graph?

Before answering the next questions, set `decay` in `LTS` back to `half-life` and the `decay constant` to 30. Now let's look at the rule for recalling items in more detail. Recall is implemented in Rule 2 of `I/O process`. Currently, this rule is set to `refracted; once`.

> **Question 6**
> What happens if you untick the `once` box in the rule's properties so that the rule is just `refracted`? Describe any differences you see in the output graph and the messages being sent from `I/O Process` to `Collate Responses`.

Question 7
What does it mean for a rule to be refracted? Set the properties of the `Recall` rule to `unrefracted`. What happens when you run the model now, and why? (To stop a simulation, click on the red square button.) Finally, modify the rule by removing the recursive call: click on the button to the left of the `send recall to I/O Process` clause, and choose `Delete this action`. Recall is now in parallel. Examine the messages sent from `I/O Process` to `Collate Responses`. Now make the rule `refracted` and examine the messages again. What is the difference, and why? Which version do you think is more cognitively plausible?

5 Extending the Modal Model

You shouldn't modify the original version of the model. Instead go to the research program manager and create a copy of the original modal model. Work on this copy for the rest of this section.

Your task is to extend the model to simulate the effect of neurological impairment. The hippocampus is a brain region that is involved in memory, especially in transferring information from short term memory to long term memory. Patients with damage to the hippocampus (e.g., Alzheimer's patients) have problems forming new memories.

Damage to the hippocampus can be modeled by modifying the element `Rehearsal`, which currently simply transfers items from `STS` to `LTS`. Introduce a new buffer `Forgotten` that contains items that have been forgotten; no read access should be possible from this buffer. Now add new rules and conditions to the `Rehearsal` process so that it randomly sends items to `Forgotten` instead of to `LTS` (make sure that items don't end up in both buffers). You will need to define a condition that generates a random number using the `arithmetic/random number` condition type.

Question 8 (more challenging)
Give the box-and-arrow diagram of the new model (subject part only) and list the rules and conditions for your new version of `Rehearsal`. What is the recall graph you get now if you run the model 10 times?

Question 9
Discuss the cognitive plausibility of the new model that you have developed. Sketch an alternatives approach to modeling hippocampal damage that doesn't require a `Forgotten` buffer.

Literature

Cooper, Richard P. 2002. *Modelling High-Level Cognitive Processes.* Mahwah, NJ: Lawrence Erlbaum Associates.