

# The COGENT Tutorial

22<sup>nd</sup> Annual Conference of  
the Cognitive Science Society

August 12<sup>th</sup>, 2000

Philadelphia, USA

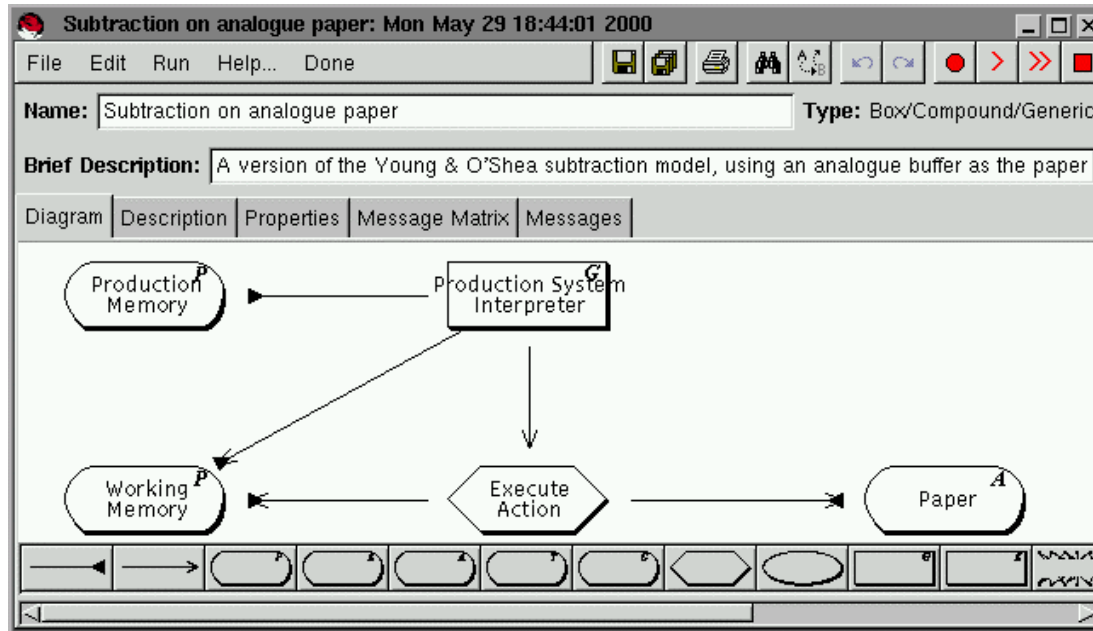
# Tutorial Overview

- COGENT: An Overview
- The Tutorial Task: The Modal Model of Memory
- The COGENT ‘Modal Model’ Model
  
- Building the Short-Term Store
- Adding the Long-Term Store
- Decay, Time and Rehearsal

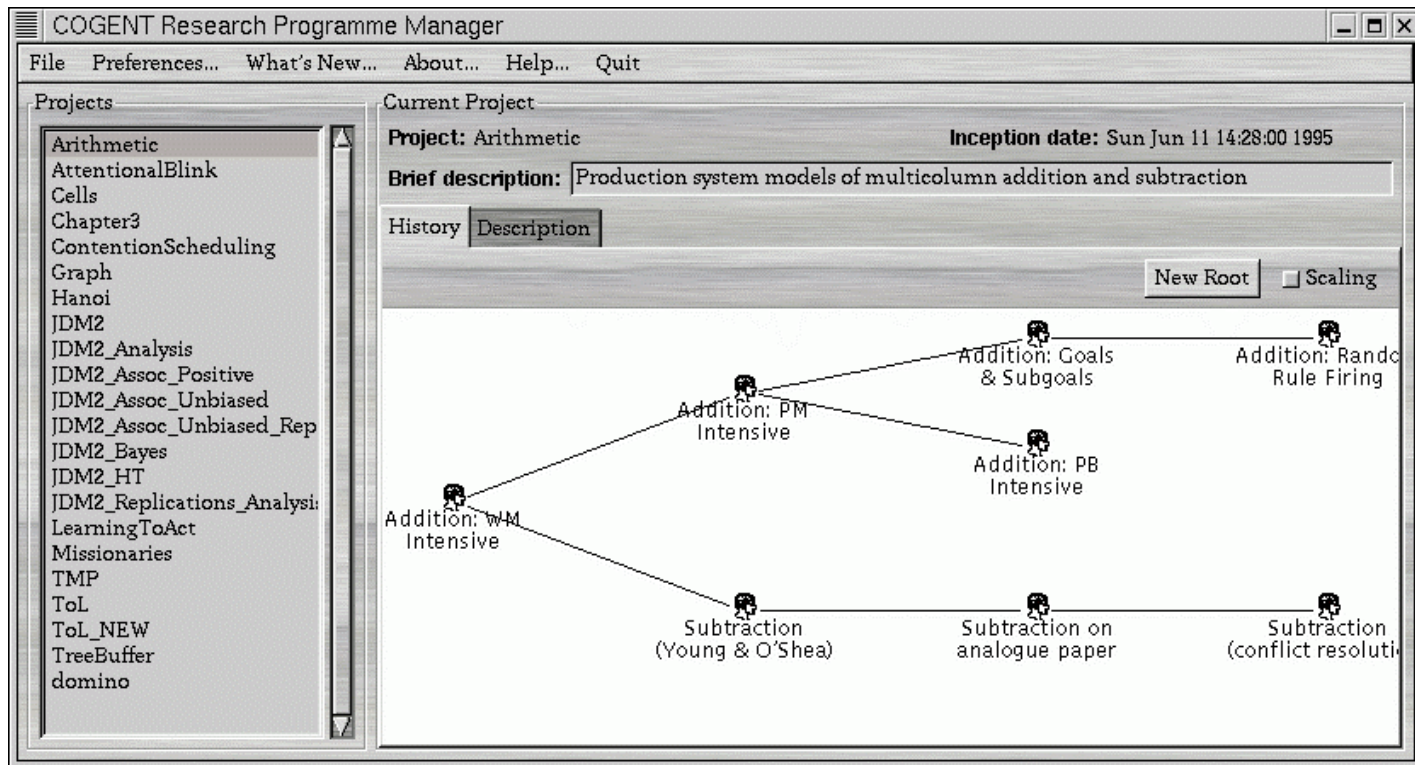
# COGENT: Principal Features

- A visual programming environment;
- Research programme management tools;
- A range of standard functional components;
- An expressive rule-based modelling language and implementation system;
- Automated data visualisation tools; and
- A powerful model testing environment.

# Visual Programming in COGENT



# Research Programme Management



# Standard Functional Components

- A library of components is supplied:
  - Memory buffers
  - Rule-based processes
  - Simple connectionist networks
  - Data input/output devices
- Components can be configured for different applications

# Rule-Based Modelling Language: I

Processes may contain rules such as:

IF        minuend(X) is in *Working Memory*  
          subtrahend(X) is in *Working Memory*

THEN add equal(minuend, subtrahend) to *Working Memory*  
      send difference(0) to *Write Answer*

# Rule-Based Modelling Language: II

COGENT's representation language is based on Prolog:

IF      $\text{minuend}(X)$  is in *Working Memory*  
        $\text{subtrahend}(X)$  is in *Working Memory*

THEN add  $\text{equal}(\text{minuend}, \text{subtrahend})$  to *Working Memory*  
       send  $\text{difference}(0)$  to *Write Answer*



# Rule-Based Modelling Language: III

Match Productions: Fri Jun 9 12:56:22 2000

File Edit Run Help... Done

Name: Match Productions Type: Box/Process

Brief Description: Update match memory on quiescence with all production matches

Rules & Condition Definitions | Description | Properties | Messages

**Rule 1 (unrefracted):** *Add new matching productions to match memory*  
TRIGGER: system\_quiescent  
IF: rule(R, C, A) is in Production Memory  
    match\_conditions\_in\_wm(C, Matches)  
THEN: add rule(R, C, A, Matches) to Match Memory

**Condition Definition:** *match\_conditions\_in\_wm/2: Match the LHS of a production*  
match\_conditions\_in\_wm([], []).  
match\_conditions\_in\_wm([HIT], [HIMatches]) :-  
    H is in Working Memory  
    match\_conditions\_in\_wm(T, Matches).

Data Element If ... then ... f(X):-g(X) Comment Summarise

# Data Visualisation Tools: Tables

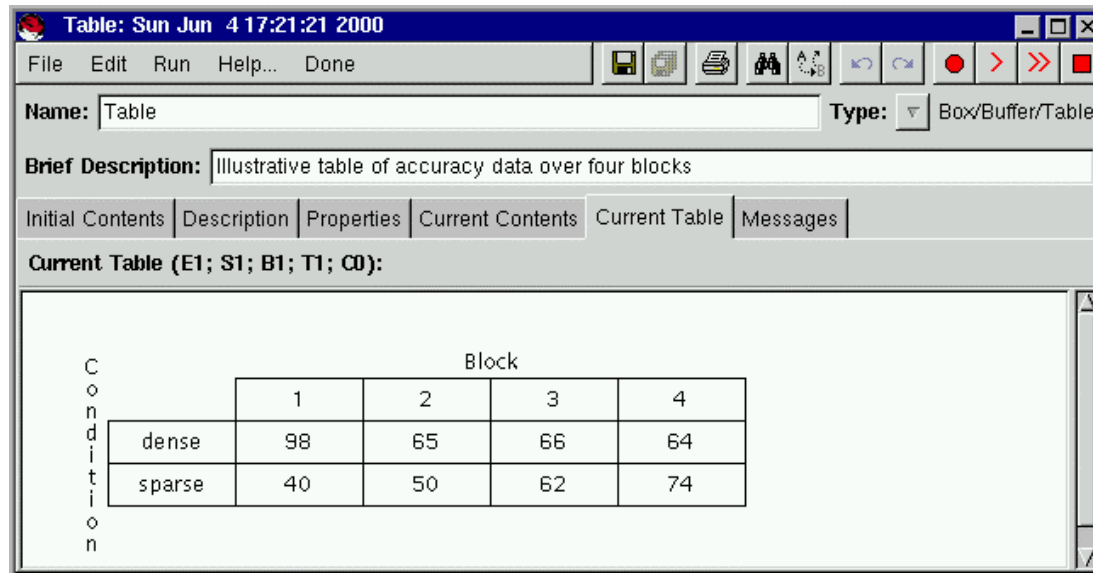


Table: Sun Jun 4 17:21:21 2000

File Edit Run Help... Done

Name: Table Type: Box/Buffer/Table

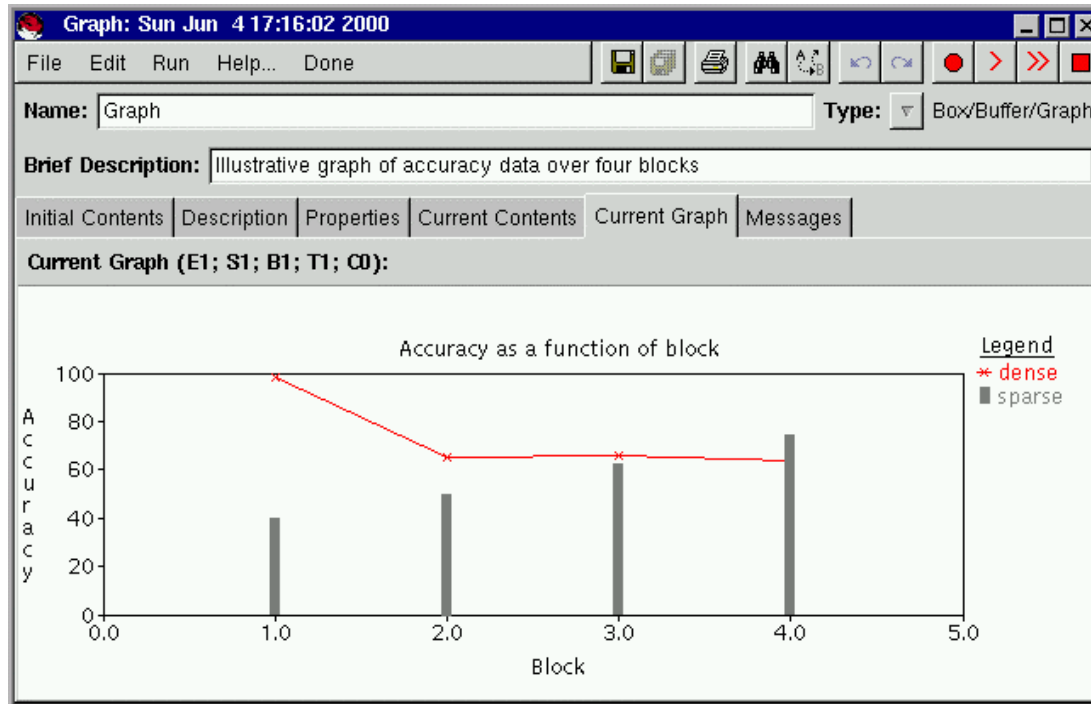
Brief Description: Illustrative table of accuracy data over four blocks

Initial Contents Description Properties Current Contents Current Table Messages

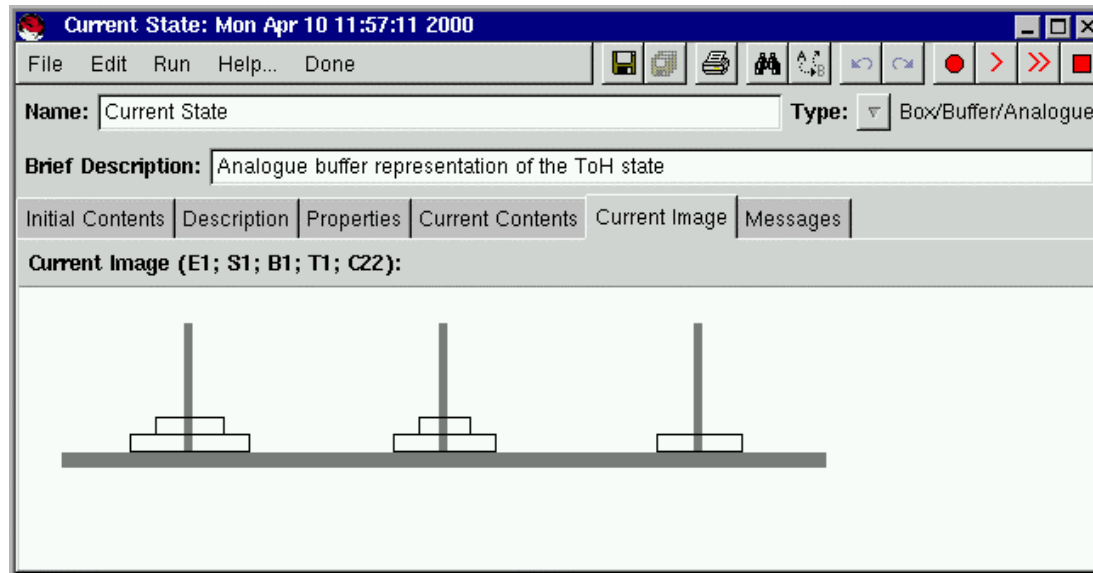
Current Table (E1; S1; B1; T1; C0):

		Block			
C o n d i t i o n		1	2	3	4
dense		98	65	66	64
sparse		40	50	62	74

# Data Visualisation Tools: Graphs



# Data Visualisation Tools: Pictures



# The Model Testing Environment

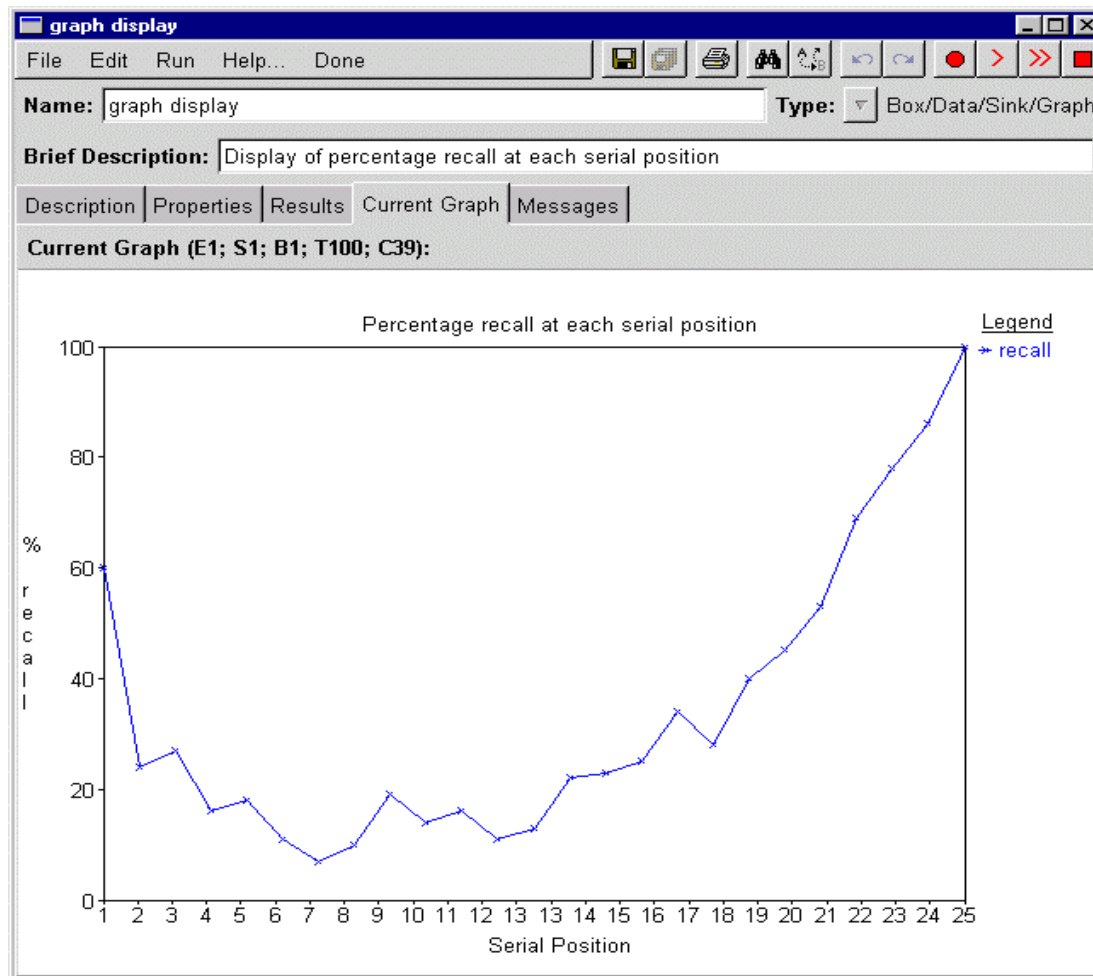
- Visualisation tools are dynamically updated
- Facilities are included to trace inter-component communication
- A flexible “scripting” environment allows:
  - models to be run over multiple blocks of trials;
  - multiple “subjects” to be run over multiple blocks;
  - automated parameter varying “meta-experiments”.

# The Tutorial Task: Free Recall

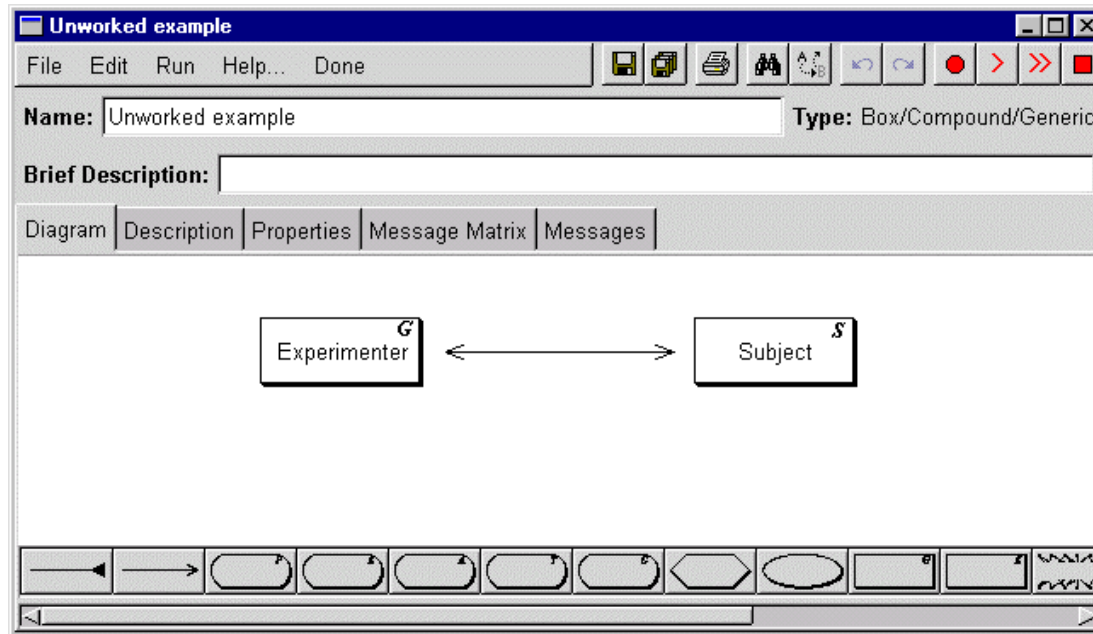
- On each trial, the subject is presented with a list of 25 words
- The subject is told to try to memorise the words
- After an interval, the subject must recall as many words as possible

(Glanzer & Cunitz, 1966)

# Free Recall: Empirical Findings

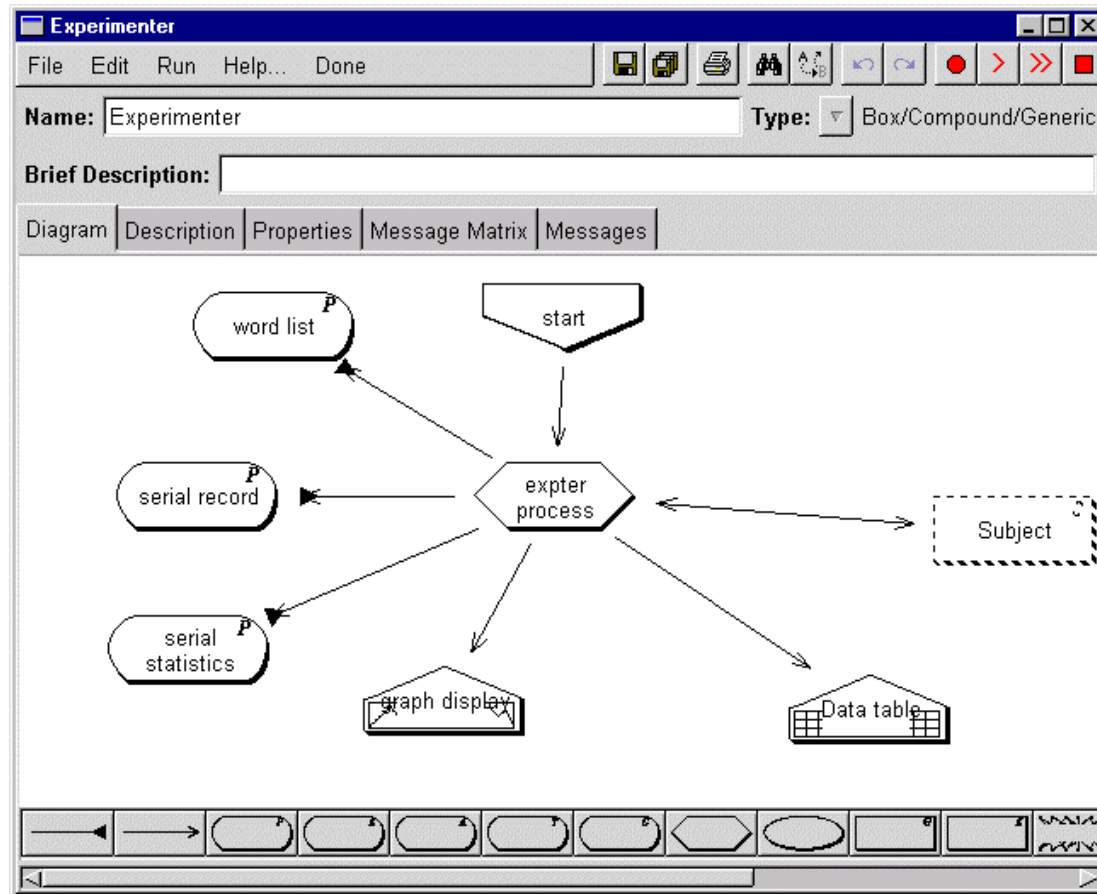


# The Modal Model: Top Level

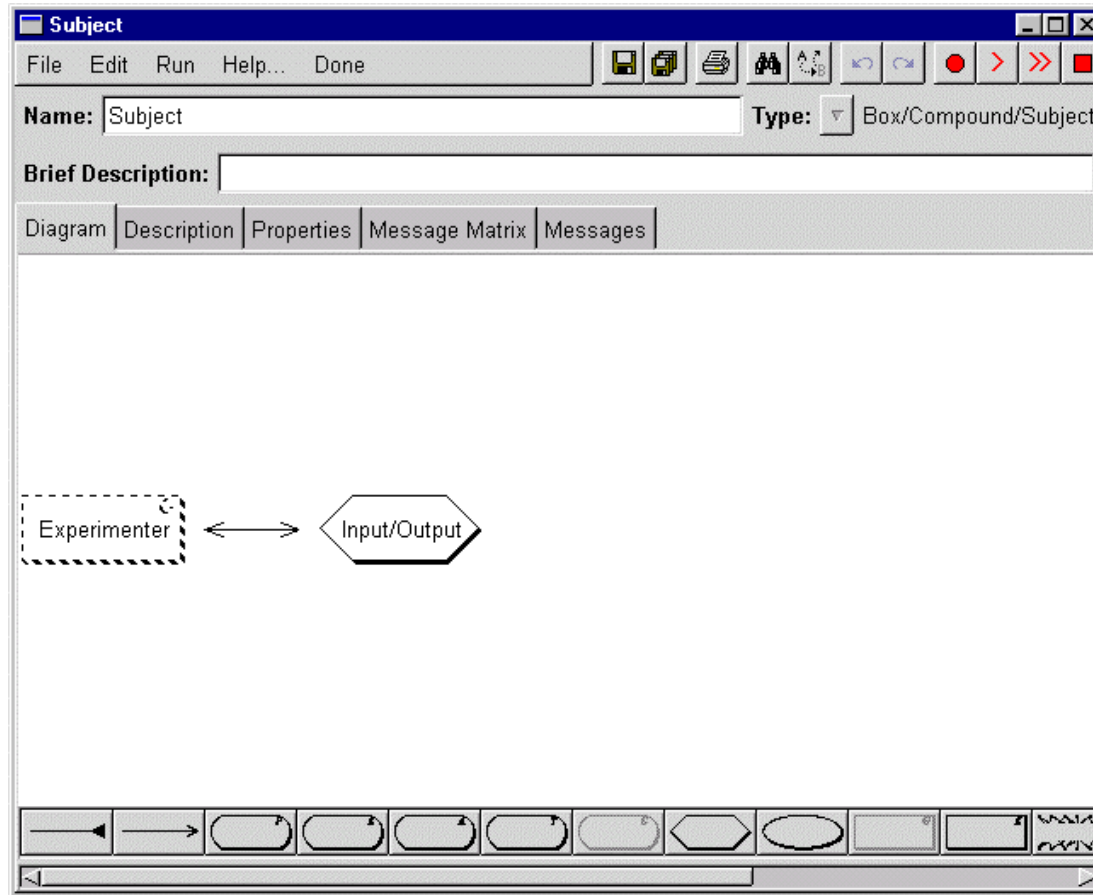




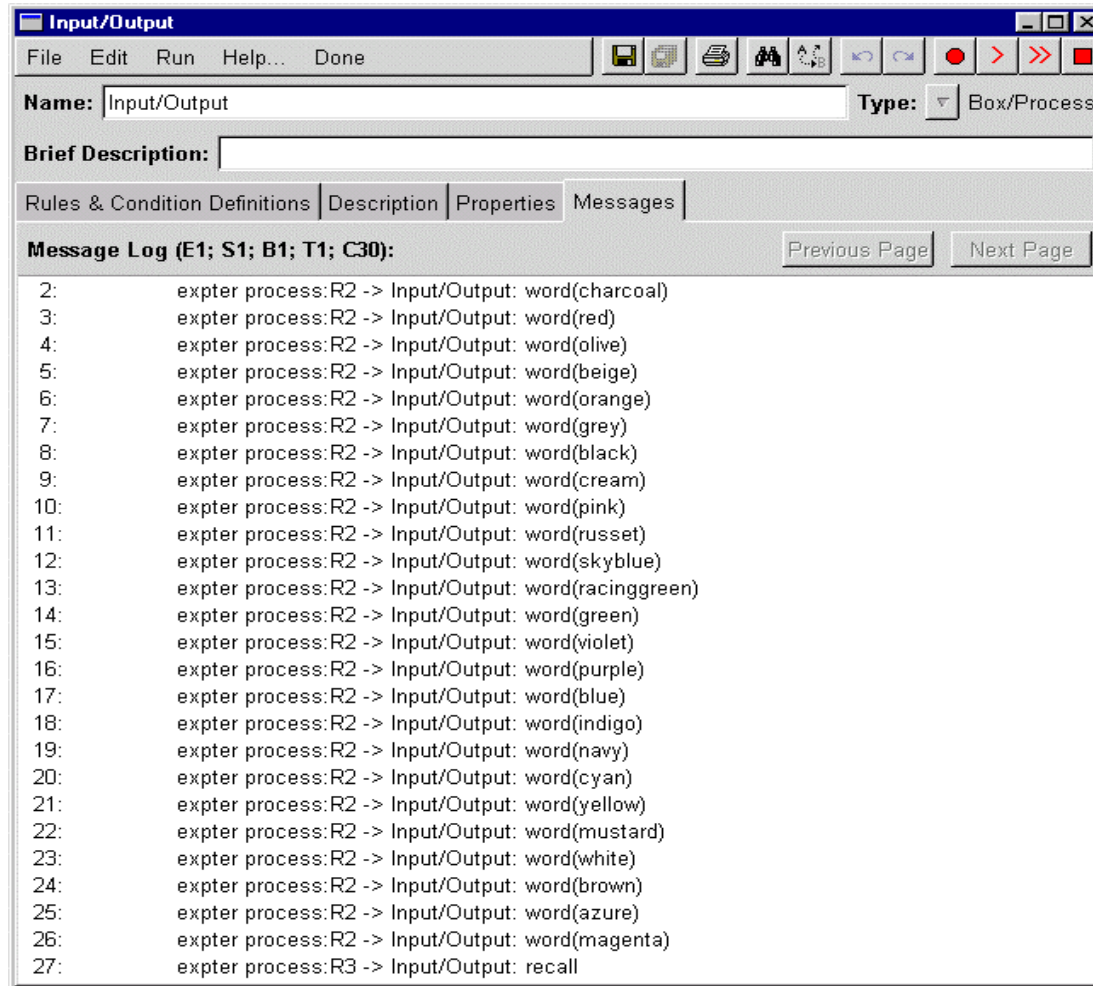
# The Modal Model: *Experimenter*



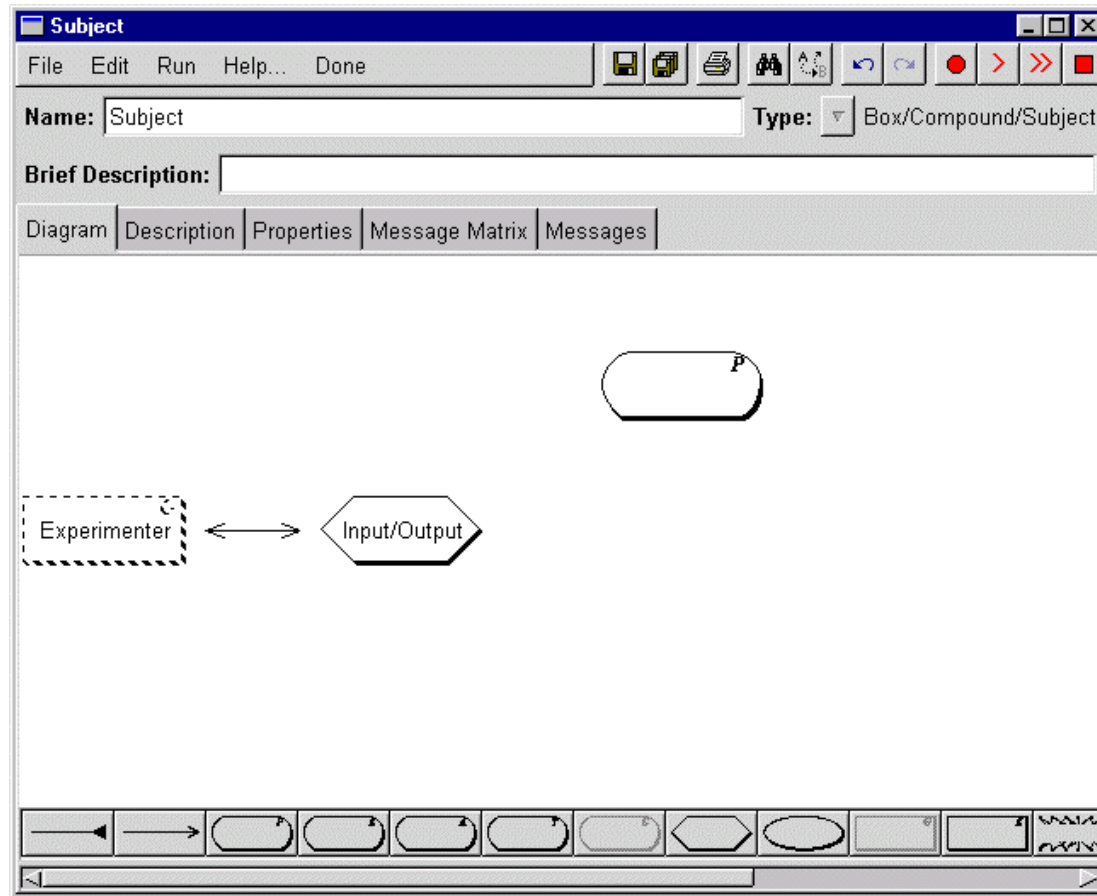
# The Modal Model: *Subject*



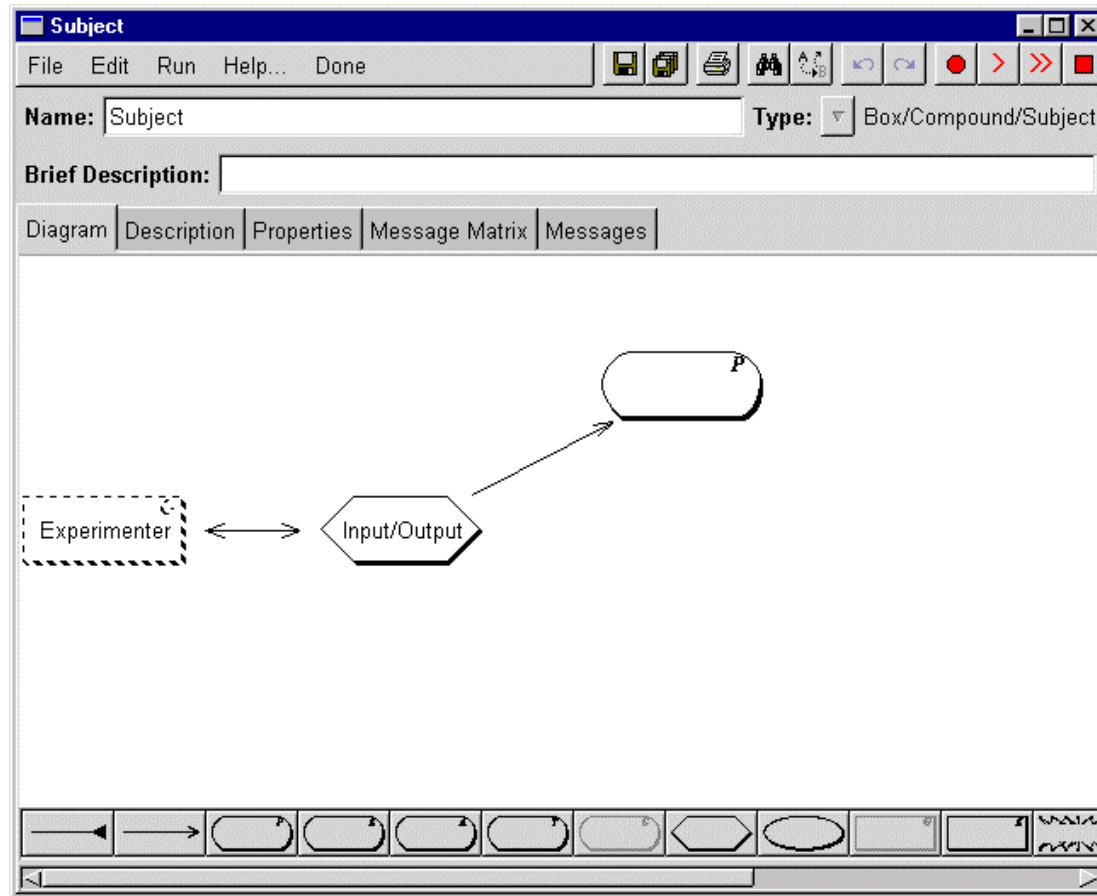
# The Model Model: Messages



# Building the Short Term Store: I



# Building the Short Term Store: II



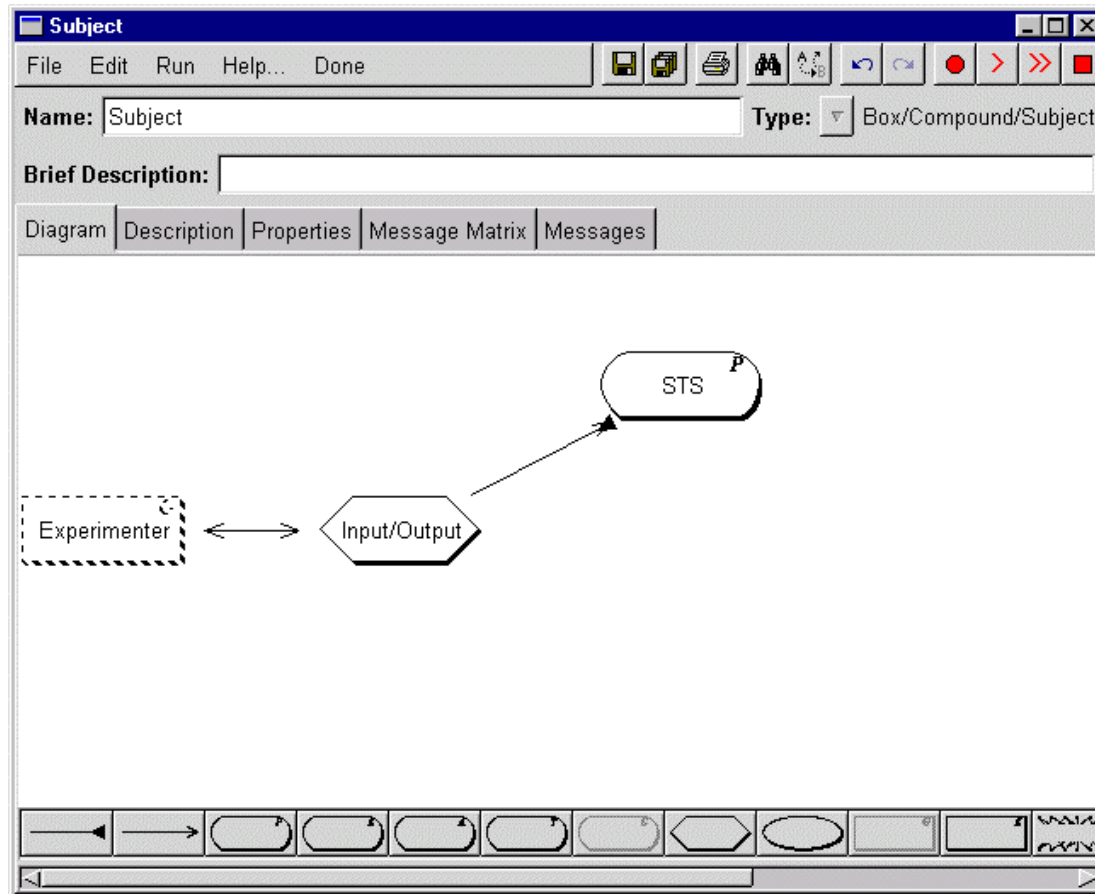
# Building the Short Term Store: III

The rule to transfer words to *STS*:

The screenshot shows a window titled "Subobject Editor: Item 1 of Input/Output\*". The window contains the following elements:

- Comment:** A text field containing "Store incoming words in STS" and a "Done" button.
- Options:** Three checkboxes: "Rule is triggered?" (checked), "Rule is refracted?" (unchecked), and "Rule fires once per cycle?" (unchecked).
- Triggering pattern:** A text field containing "word(X)".
- Conditions:** A large empty text area with an "Add Condition" button.
- Actions:** A text area containing "add X" in a dropdown menu, "to", and "STS" in another dropdown menu, with an "Add Action" button.

# Building the Short Term Store: IV





# Building the Short Term Store: V

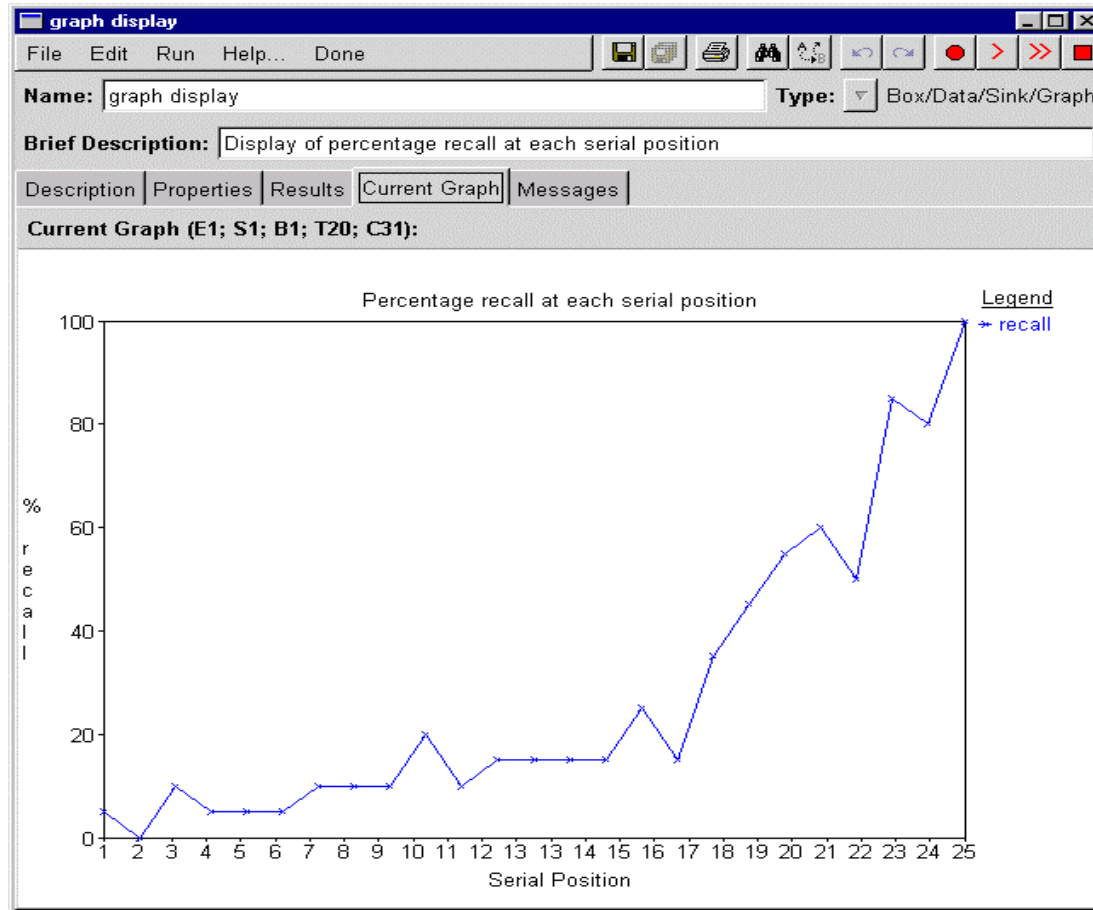
The rule to recall from *STS*:

The screenshot shows a window titled "Subobject Editor: Item 2 of Input/Output\*". The window contains the following elements:

- Comment:** A text field containing "The recall rule" and a "Done" button.
- Options:** Three checkboxes: "Rule is triggered?" (checked), "Rule is refracted?" (unchecked), and "Rule fires once per cycle?" (unchecked).
- Triggering pattern:** A text field containing "recall".
- Conditions:** A section with an "Add Condition" button. It contains a condition: a dropdown menu with "Word" selected, followed by "is in", another dropdown menu with "STS" selected.
- Actions:** A section with an "Add Action" button. It contains an action: a dropdown menu with "send" selected, followed by a text field containing "recalled(Word)", followed by "to", another dropdown menu with "exptor process" selected.



# Building the Short Term Store: VI



# Building the Short Term Store: VII

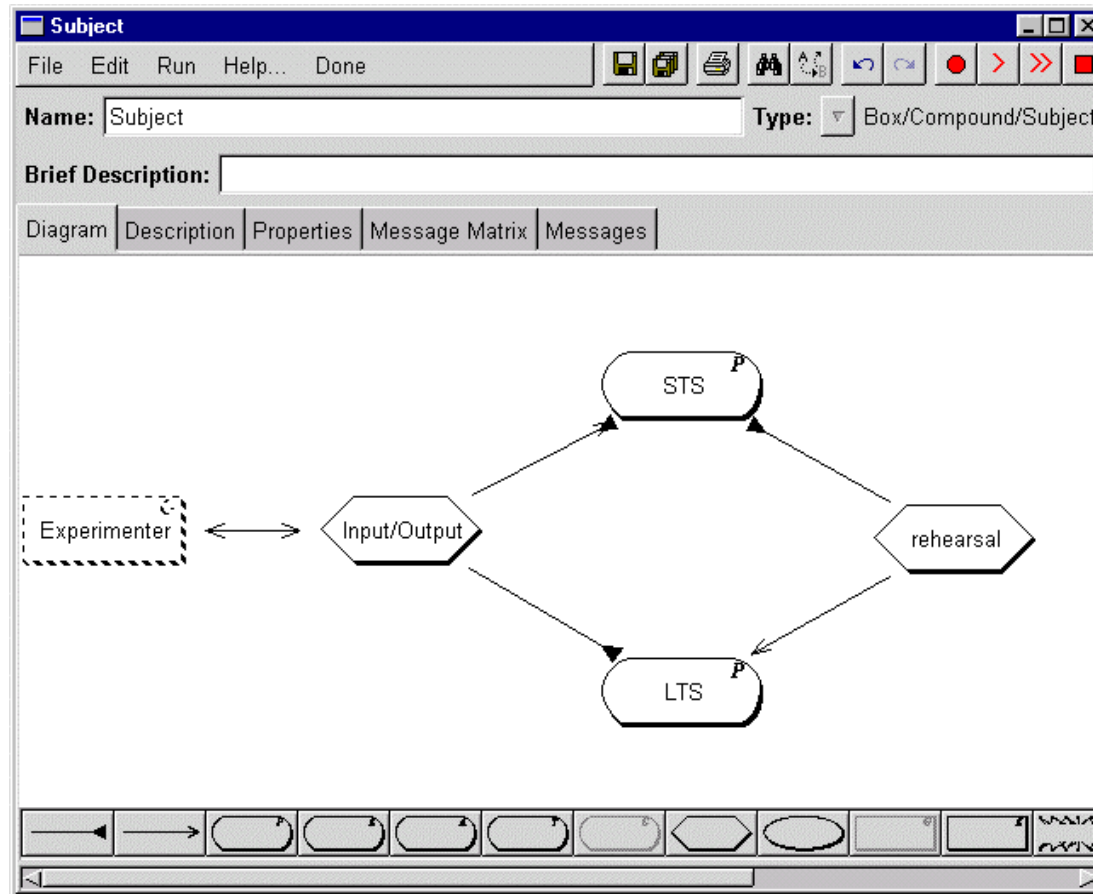
- Run more trials. What happens to the curve?
- Change the On Excess property of *STS*. What happens to the shape of the graph when you run a few trials?
- Watch the Messages view of *Input/Output*. What happens there now when you run (or single-step) through a trial?

# Adding the Long Term Store: I

The modal model also includes:

- a long term store (LTS);
- a rehearsal process to transfer information from STS to LTS; and
- the possibility to recall from either STS or LTS

# Adding the Long Term Store: II



# Adding the Long Term Store: III

The rehearsal rule:

**Subobject Editor: Item 1 of rehearsal** [min] [max] [close]

**Comment:** Limited-capacity rehearsal channel [Done]

Rule is triggered?       Rule is refracted?       Rule fires once per cycle?

Triggering pattern: [ ]

**Conditions:** [Add Condition]

[ ] Word is in [ ] STS

**Actions:** [Add Action]

[ ] add Word to [ ] LTS

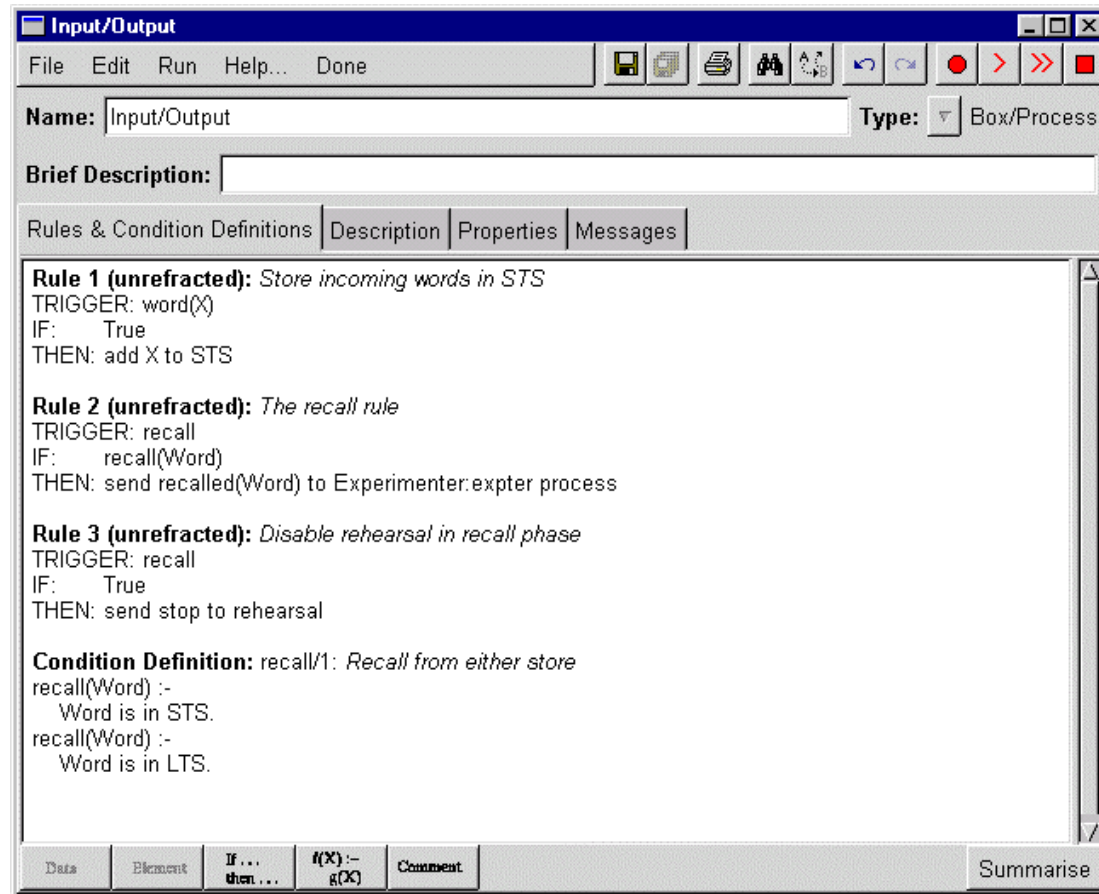
# Adding the Long Term Store: IV

Recalling from either *STS* or *LTS*:

The screenshot shows a window titled "Subobject Editor: Item 4 of Input/Output\*". The window contains the following elements:

- Comment:** A text field containing "Recall from either store" and a "Done" button.
- Functor:** A text field containing "recall".
- No. of args:** A spin box set to "1" and an "Add Clause" button.
- Clause List:** A list of two clauses, each with a collapse/expand icon on the left:
  - Clause 1: "recall ( Word ) :-" followed by a sub-clause "Word is in STS".
  - Clause 2: "recall ( Word ) :-" followed by a sub-clause "Word is in LTS".

# Adding the Long Term Store: V



# Adding the Long Term Store: VI

- What causes the Primacy Effect arise?
- Monitor the *Input/Output* box's Messages view. Why does the model sometimes recalls the same word twice in the same trial.
- The serial position curve still doesn't look like the one in the introduction. Characterise the difference. Can you account for it?



# Decay, Time & Rehearsal: I

- Add decay to *LTS*. Explore different decay rates.
- Change the rehearsal rate by adding a copy of the rehearsal rule.
- All memorised words are currently recalled in parallel. Make the recall process serial.

# Decay, Time & Rehearsal: II

The serial recall rule:

**Subobject Editor: Item 2 of Input/Output**

**Comment:** The recall rule Done

Rule is triggered?       Rule is refracted?       Rule fires once per cycle?

Triggers pattern: recall

**Conditions:** Add Condition

recall ( Word )

**Actions:** Add Action

send recalled(Word) to Experimenter:expt process

send recall to Input/Output

# Decay, Time & Rehearsal: III

- Explore the effect of the Buffer Access property of each buffer. Play with these (and other) parameters to see how they affect the model's behaviour.
- The Experimenter system is written using standard COGENT. Try to discover how it works.
- Go on to develop the model into something substantial.

# References

- Atkinson, R. C., & Shiffrin, R. M. (1968). Human memory: A proposed system and its control processes. In Spence, K. W., & Spence, J. T. (Eds.), *The psychology of learning and motivation: Advances in research and theory*. Academic Press, Orlando, FL.
- Atkinson, R. C., & Shiffrin, R. M. (1971). The control of short term memory. *Scientific American*, 225, 82–90.
- Cooper, R., & Fox, J. (1998). COGENT: A visual design environment for cognitive modelling. *Behavior Research Methods, Instruments, & Computers*, 30(4), 553–564.
- Glanzer, M., & Cunitz, A. R. (1966). Two storage mechanisms in free recall. *Journal of Verbal Learning and Verbal Behavior*, 5, 351–360.
- Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63, 81–97.
- Young, R. M., & O’Shea, T. (1981). Errors in Children’s Subtraction. *Cognitive Science*, 5, 153–177.