

AD-A257 180



CAPS ReReader Simulation Model

User's Manual

Version 1.0

Julio Ortega, Elizabeth U. Saul, Sashank Varma, & Susan R. Goldman

Learning Technology Center
Vanderbilt University

October, 1992

100-443887-1000

ALL INFORMATION CONTAINED HEREIN IS UNCLASSIFIED
DATE 10-10-2001 BY 60322 UCBAW

The development of the system described in this report was supported by the Office of Naval Research Cognitive Science Program 1142CS Grant N00014-91-J-1769, and in part by the National Science Foundation, Grant DBS 90-09320.

92-28716



608

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE 10/29/92	3. REPORT TYPE AND DATES COVERED Technical Manual 5/1/91 - 4/30/92	
4. TITLE AND SUBTITLE CAPS ReReader Simulation Model User's Manual Version 1.0			5. FUNDING NUMBERS Grant N00014-91-J-1769	
6. AUTHOR(S) Julio Ortega, Elizabeth U. Saul, Sashank Varma, & Susan R. Goldman				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Learning Technology Center Vanderbilt University Box 45, Peabody Nashville, TN 37203			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Cognitive Science Program (1142CS) Office of Naval Research 800 N. Quincy Street Arlington, VA 22217-5000			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Unrestricted			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) CAPS ReReader is a computer program that simulates the process and products of reading comprehension. The input to ReReader is a representation of a text plus the representation of some minimal knowledge that a human would need for the comprehension of the text. A simulation run of ReReader produces traces of the reading behavior of a hypothetical reader and outputs a set of "long term memory" strengths for the text information. The results of the traces can be analyzed and compared with reading behavior and recall data collected from human subjects reading the same text. ReReader currently runs on a VAX workstation running under the VMS operating system. ReReader Version 1.0 uses two User interfaces: One permits the User to set the parameters for a simulation run; the other is for specifying different data report forms. This User Manual has four sections plus appendices of technical information: Section 1 describes ReReader's text processing model. Section 2 explains how to run a ReReader simulation. Section 3 explains how to obtain and interpret the output of a ReReader simulation with the help of the ReReader Report interface. Advanced features of the ReReader simulator are described in section four.				
14. SUBJECT TERMS Computational modeling Text Comprehension			15. NUMBER OF PAGES 59	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT	

Table of Contents

User's Manual Overview	1
1. The ReReader model	2
2. The User Interface for ReReader	9
3. Getting and Interpreting ReReader Results	13
4. Advanced Features	18
Appendices:	
A: Login and logout of the ReReader account	25
B: Naming, Creating, and Transferring Files	26
C: Simulation File Formats	29
D: MOTIF User Interface Components	34
References	37
Tables and Figures	38

Accession For	
NTIS	☒
DTIC	☐
USCIB	☐
J. 101	
By	
Date	
Approved by	
Dist	Accession for Special
A-1	

DIRECTOR, ARMY RESEARCH OFFICE

User Manual Overview

CAPS ReReader is a computer program that simulates the process and products of reading comprehension. The input to ReReader is a representation of a text plus the representation of some minimal knowledge that a human would need for the comprehension of the text. Once the input is prepared and the simulation is run on that input, ReReader produces traces of the reading behavior of a hypothetical reader. As well, ReReader outputs a set of "long term memory" strengths for the text information; these are systematically related to the way the hypothetical reader read the text. The results of these traces can be analyzed and compared with reading behavior and recall data collected from human subjects reading the same text.

The present implementation of ReReader provides two interfaces that facilitate the tasks mentioned above. One helps run the simulation and the other helps analyze the results of the simulation. This manual describes how to use these interfaces. ReReader currently runs on a VAX workstation running under the VMS operating system (See Appendix B for details).

The first section of this document describes the ReReader model of text processing. The second section explains how to prepare the input and how to run a ReReader simulation using the simulation interface. The third section explains how to obtain and interpret the output of a ReReader simulation with the help of the ReReader report interface. The fourth section explains advanced features of the ReReader simulator. Finally, a series of appendices provide further details on procedures and features of the ReReader system and its computer environment.

1. The ReReader Model

The ReReader simulator is a computational implementation of the strategic competition model of text comprehension of Goldman and Saul (1990). The ReReader simulator was implemented in the Collaborative Activation-based Production System (CAPS) architecture of Just and Carpenter (1992). CAPS provides the basic mechanisms to model both working memory and strategy competition aspects of text comprehension.

1.1 The CAPS Architecture

The CAPS architecture permits one to build "comprehenders" that have (i) different knowledge but the same "working memory resource" capacities, or (ii) different capacities but the same knowledge. By manipulating each of these independently, it is possible to observe the independent and joint effects of each factor on comprehension processes and on the products of those processes.

A key aspect of the CAPS architecture is that preconditions on actions can exist at *levels of activation* rather than as merely present or absent. So a "reader model" may behave differently depending on the other elements in working memory, the activation limit for working memory, and the available activation. In the CAPS architecture when the activation limit is reached, information already in working memory has to share the activation with new elements that enter working memory; the "older" elements lose activation to the newer ones. When elements fall below a minimum level, they are no longer functional in working memory; they cannot connect with new information. They are, for all intents and purposes, gone.

1.2 Characteristics of The ReReader Model

The ReReader System, outlined below, is a prototype model and is limited in scope. The present implementation of ReReader focuses on establishing connections

among propositionalized sentences and on the repair strategies used when initial proposition-linking procedures fail. The user can manipulate the characteristics of the "reader" through a user-friendly interface. Elaborations of ReReader that embellish the knowledge base and the way strategy competition operates are under development.

1.2.1 Representation of the input text.

The system operates on a propositional text base. A sample is given in Table 1 using the first sentence of the passage shown in Table 2. Sentences are parsed into a verbal predicate with optional and obligatory arguments (predicate 1) along with the relevant concepts (existence nodes), and modification propositions. A fourth type of proposition relates predicates, e.g., cause, contrast, etc. In ReReader the propositions are connected using three linking strategies: argument overlap, propositional embedding, and relational reference. There are several variables that can be adjusted: (a) The weights, or initial activation values, assigned to the propositions; (b) The linking strategies the system operates with; and (c) The weights assigned to the links. The four types of propositions that ReReader uses - verbal predicates, concept propositions, modification propositions, and relational predicates - are assigned default weights of 2 (verbal predicate) and 1 (everything else). Table 1 also shows the default weights for each type of proposition. Links between propositions are weighted equally (a value of 1). The activation values for propositions and links may be modified through the user interface.

1.2.2 System Structure

There are two kinds of "spaces" in the system: one corresponds to human working memory and the other to long term memory. Incoming input "occupies" working memory and processing uses working memory capacity. The system has in long term memory a dictionary that contains information classifying the predicates for the specific text, a set of equivalence elements (translators), and a set of

procedural elements such as the propositional linking strategies and reading strategies.

The variables related to system knowledge are the following: the Activation Cap, i.e., the capacity of working memory; the amount of prior knowledge in the dictionary; procedural knowledge available to the system; and motivation level, (how motivation affects the behavior of the production system is described below).

When working memory capacity is exhausted (the Activation Cap is reached), a proportionate degradation of the activation of all elements active in working memory occurs. The Activation Cap for each run is set through the interface (we have experimented with Caps in the 30 to 125 unit range thus far). Different Caps will affect the life of an input proposition in working memory because degradation of working memory elements occurs when the Cap is reached. For example, when reading the second sentence of a passage, the working memory resources might be sufficient to handle creation of all requested elements at the targeted initial activations without stealing activation from elements preexisting in working memory. On the fourth sentence, however, all available activation might be allocated and further requests, perhaps to build within-sentence links, would result in the degradation of all elements (i.e., their activations would be scaled back). Over many processing cycles at the Cap, a working memory element would lose most of its activation and fall below threshold, thus becoming invisible (unusable) in further processing unless it is re-activated.

1.2.3. Reading Strategies

The system "knows" four reading strategies: Read forward, read backwards, skim backwards to a problem, skim forward to sentence that initiated a reread. Skim differs from "read" in that orthographic representations of words get low levels of activation or reactivation; during read propositional linking is done and some propositions may be reentered into active states in working memory. The four strategies are based on the behavioral data of Goldman and Saul (1990).

The strategies specified in the Strategy Competition Model (Goldman & Saul, 1990) involved different numbers of sentences: Sometimes a reader read just one sentence back, sometimes three or four sentences back to a specific sentence. In ReReader, these different kinds of strategies are a result of evaluations of the current strategy after the processing of each sentence. These evaluations depend on several aspects of coherence and the motivation level of the system. The evaluation function is described more completely below.

1.2.4. How ReReader Works.

As illustrated in Figure 3, the ReReader system operates in four conceptual phases as follows:

Phase I. Reads the first sentence as a string of propositions and assigns activation levels according to the (default) framework described in Table 1: 1 for concepts, modification, and relational propositions and 2 for main predicates of the sentence. If the working memory "cap" or limit is exceeded, elements in working memory "lose" activation proportionally until the new elements can realize the appropriate values to be recognized by the system.

Phase II. Creates within sentence links. For the sentence shown in Table 1, P1 would be connected to P4 and to P5.

Phase III, between sentence links are created according to argument overlap, propositional embedding and relational reference rules.

Phase IV. This phase proceeds in two steps. First, ReReader computes measures of text coherence and summarizes them with an evaluation function, as explained in subsection 1.2.5. The second step is to evaluate the current reading strategy, using the measures of text coherence as inputs, and returning the "next" reading strategy. "Read forward" will be selected if there is a positive value returned by the evaluation function. "Mark problem" will be selected if there is a zero value returned by the evaluation function. "Read backwards" will be selected if there is a negative value returned by the evaluation function. How the system proceeds given

each of these outcomes is described later on, in subsection 1.2.6

1.2.5 Evaluating Text Coherence

Text Coherence is computed using the following five pieces of information:

1. Number of links to the first sentence of the passage if the current sentence is the first sentence of a paragraph. A higher value equals greater coherence.
2. Number of links to the first sentence of the paragraph if the current sentence is not the first sentence of a paragraph. A higher value equals greater coherence.
3. Number of links to previous sentence (superseded by 2 or 1). So if the current sentence is second in the paragraph, procedure 2 will operate and 3 will not. For the third sentence in a paragraph, procedure 3 will operate as will procedure 2 but they will be counting different links.
4. Number of propositional arguments referenced in the current sentence that are still present in working memory.
5. Number of propositional arguments referenced in the current sentence that are missing from working memory. This measure reflects problems integrating the current input with elements already in working memory; in other words, failure to find overlap.

These five measures of text coherence are combined using an evaluation function. The formula for the Evaluation Function reflects competition between reasons for believing adequate coherence exists and reasons for believing it does not, with motivation weighting the latter. More precisely, it is the sum of measure 1 (above) plus measure 2 plus (measure 3 divided by 5) less (measure 5 + motivation).

$$EF = \text{Msr. 1} + \text{Msr. 2} + (\text{Msr. 3}/5) - (\text{Msr. 5} + \text{Motivation})$$

The reason measure 3 is divided by 5 is so that local links will not totally dominate reading. The denominator of 5 allows the system to take into account more global aspects of coherence as well as coherence with the just prior sentence.

Motivation has a direct effect on the likelihood of rereading. With high

motivation going back will be more likely than if motivation is low. In fact, with this evaluation function, two readers with the same number of missing arguments would behave differently depending on their motivation levels. (Note that measure 4, the number of propositional arguments active in working memory, does not figure in the evaluation function. We spent two weeks searching for a suitable function, one which seemed plausible and produced reasonable behavior. Although the present evaluation function does not include measure 4, replacing measure 5 in the evaluation function given above the proportion of missing arguments, given by $\text{Msr. 5} \div (\text{Msr. 4} + \text{Msr. 5})$ yields performance quite similar to that obtained with the simpler evaluation function.)

1.2.6 Competition among Reading Strategies

As Figure 2 illustrates, strategy selection depends on the evaluation function in the following manner. If the evaluation function returns a positive or zero value strategy selection is relatively trivial. A positive value means that there is sufficient connection between the current sentence and propositions still active in working memory. In other words, coherence is okay so reading continues forward. A zero value indicates that the sentence is partially connected but that there are a number of potential connections that were not made. The sentence will be marked as a problem. If sentences are marked, the marker becomes important if the system decides to read backwards. In the future, we intend to use these kinds of markers for decisions at the end of text about general level of understanding and comprehension.

A negative value results in the system reading backwards. This is essentially a judgment that the current sentence lacks coherence with the propositions active in working memory. If the system decides to read backwards, there are two possibilities: it can skim back to a marked-problem sentence or it can read backward sequentially from the current sentence.

a. The Marked-problem case: This case occurs when there is a proposition active in working memory that has been marked as a problem. (These will tend to be within 3 or 4 sentences back for most normal activation caps.) The system skims back to that sentence, reads it and then evaluates. If the evaluation function returns a positive value or a zero value the problem was resolved sufficiently. ReReader skims forward to the sentence that initiated the rereading and continues reading forward (i.e., reads in the next sentence). However, if the evaluation function returns a negative value, the problem remains; the system then proceeds backwards from the marked problem sentence. It stops going backwards when the evaluate function returns read forward. (Note that in this case, ReReader skips over sentences between the marked problem sentence and the sentence that initiated the rereading. This is reasonable because in most cases if those "skipped over" sentences had had a lot of coherence with the sentence that initiated the rereading there would not have been a coherence problem to begin with, i.e., the "skipped over" propositions were available for integration the first time around.)

If there are multiple, marked sentences active in working memory, ReReader goes through them sequentially in the following way: If there are multiple marked problems in memory, go to nearest one and read it. Evaluate. If there is still lack of coherence and there are more marked problems go to the next one. Evaluate. If coherence is still lacking and there are no more marked problems, read backwards.

b. Read backward sequentially. This case occurs when there are no or no more "marked-problem" sentences in working memory. After each sentence is read, there is an evaluation. When Evaluate does not return read backwards, skim forward again to sentence that initiated the backtrack and read the next sentence.

In cases of read backwards, if the first sentence of the passage is reached and the evaluation function is still returning read backwards, ReReader skims back to the sentence that initiated the rereading, marks it as a problem, and resumes reading forward. One additional special case may occur: During rereading it is possible for the ReReader to "lose it's starting place" (initiating sentence). If it does, it

begins reading forward from the current sentence.

In summary, the evaluation function and rereading work in the following way: ReReader reads forward and at the end of each sentence evaluates how well it integrates with information active in working memory. If there are small problems, ReReader marks the sentence and continues reading forward. If there are larger problems, ReReader proceeds backwards. If any sentences are marked, ReReader skims back and reads each one starting at the closest one. If none are marked, ReReader reads backwards sequentially. When the evaluation function no longer returns read backwards, the problem has been solved; and ReReader skims forward to where the problem occurred and resumes reading forward.

2. The User Interface for ReReader

The ReReader *simulator* attempts to model a wide range of human behavior by permitting the researcher to control a fairly large set of parameters. The most relevant of these parameters and options can be set quite easily through the ReReader initialization interface. The window for that interface is shown in Figure 3. The basic operations needed to run the simulator using this interface are explained in this section. More advanced features of the simulator and the interface are described in section 4.

The User Interface is based on the MOTIF package. The elements of the MOTIF package used in our application (buttons, windows, scales, etc.) are described in Appendix D.

The User Interface for ReReader guides the user through the operation of ReReader. Running ReReader involves specifying a text file and "dictionary" for the text, starting the user interface, setting parameter values, selecting desired output files, and getting the ReReader simulation running. Each of these is briefly described here.

2.1 Specifying the text file and "dictionary"

The text and "dictionary" files can be produced by any number of word processing applications. If the files are created in a computer other than the VAX workstation where the ReReader simulator program resides, they will have to be transferred to the VAX workstation. Appendix B explains how to name files and how to transfer them from/to Mac Computers.

The text file. A specific format is required. The format is basically similar to that used frequently in propositional analysis (e.g., Kintsch, 1988; Turner & Green, 1978). The format is illustrated here for the first sentence of the distance passage that was parsed in Table 1. The illustration assumes that the text contains only the one sentence.

```
(convert-text '(  
(sentence 1)  
(start paragraph 1)  
distance is simply the space between 2 points  
(1 is (pp4,pp5)  
(2 is^between (pp5 pp6))  
(3 quant (pp6 2))  
(4 distance)  
(5 space)  
(6 points)  
))
```

The first element in the parentheses is the proposition number. This is followed by the verbal predicate for the proposition. We combine certain surface elements into one by linking them, as in *is^between*. Sentence elements appearing by themselves, as in line 8, indicate concepts. The sentence number and sentence (see lines 2 and 4

in the example) are included in the file. Line 3 indicates that this sentence starts a paragraph. Inclusion of this information makes it possible to subsequently implement different activation depending on the position of a sentence in a paragraph. Appendix C.1 describes the format of the text file (which we call the *passage* file) in more detail.

The "dictionary" . This is essentially ReReader's knowledge base. In the present implementation, it contains information about the assignment of each proposition to one of four types (predicate, relational, concept/existence, or modification). Future versions will also incorporate propositions representing lexical and schematic knowledge relevant to the passage. Appendix C.2 describes the format of the "dictionary" file (which we call the *predicate* file) in more detail.

2.2 Starting the User Interface (This section is specific to our implementation of Rereader.)

To get the User Interface running, you will first need to login to the ReReader computer account. If you don't know how to do this, please refer to Appendix A. Once you login to the ReReader account and obtain the "\$" prompt from the VAX Workstation, type "simulate." The ReReader initialization interface will start running, the window shown in Figure 3 will appear on the screen.

2.3 Setting Parameter Values

CAPS ReReader depends on a number of parameter values. The parameters that *may be* altered by the user are shown in Figure 3, which is a copy of the CAPS ReReader Initialization screen. The *Activation Cap* and *motivation* parameters are the two most important parameters in ReReader and need to be carefully selected for each simulation. The activation cap corresponds to short-term memory capacity in a human, while motivation corresponds to a simple measure of motivation in humans. Different settings of the activation cap will result in very different memory traces and reading behaviors.

2.4 Selecting Input and Output Files

The names of the files used as input and output for a simulation run need to be specified. Please refer to Appendix B to find out about what are valid file names in the VAX workstation. The Initialization screen will request the names of the text and the "dictionary" file for input. These files should already reside on the VAX workstation. The Initialization screen also requests information regarding the names of files in which trace information for the simulation run will be stored. ReReader produces two output files at the end of a simulation: the memory file, and the sentence trace file. These files are used to construct a variety of reports, as indicated in Figure 4, ReReader Report Selection menu.

2.5 Getting the ReReader simulation running

Once all the parameters are correctly set, click on the button labeled "Go On and Simulate". This will begin the ReReader simulation, which should take anywhere from 10 to 30 minutes for a 150-proposition text. Once the simulation finishes, you can proceed to run the reports in order to summarize the results of the simulation (See next section). You should wait until the simulation finishes before you issue another "simulate" command or a "report" command.

2.6 Help and Quit.

Help information can be obtained by clicking on the help buttons of the main interface windows (Figures 3 and 4), option windows (Figures 7, 8, and 9), and for the secondary windows which appear after you click on the help buttons. If you press the Quit Button (see bottom of Fig. 1), the ReReader simulation will abort.

3. Getting and Interpreting ReReader Results

3.1 Overview

Reports of results can be run by using the window shown in Figure 4. To make this window appear, you need to first login to the ReReader account (See Appendix A). Once you get the "\$" prompt from the VAX Workstation, type the command "report" and the menu shown in Figure 4 will appear. As can be seen on this menu, a variety of analyses can be requested, including a listing for each cycle as well as various summaries. As well, an automatic edit feature based on activation strength (threshold) is included. Only propositions that exceed the threshold are included in the report information. Also as indicated in Figure 4, the data are prepared in a format importable by spreadsheet programs. We have used Excel and have not tested other commercially available spreadsheet packages.

ReReader simulation results are generated by packaged "reports" that provide summaries of the simulated long term and working memory contents and reading behaviors. The ReReader *Report* interface facilitates report generation. Several parameters allow customization of the reports. The report interface window is shown in Figure 4 and is further explained in the following subsections.

There are three types of Reports that can be obtained.

a) Data sets that can be imported into other software for further analysis.

(Examples are provided below in Reports 1, 2, 3, and 6.)

b) Data that is to be read and manually interpreted by humans. (Examples are provided below in Reports 4 and 5.)

c) Graphics. (An example is provided below in Report 7.)

To select a particular report, click on the relevant line shown in the Report Interface menu (Figure 4). Only one report can be selected at the time. Make sure all the necessary parameters for the desired report have been set to the correct values. (Sections 3.2 through 3.9 describe the parameters required for the various reports.)

Clicking the "Go Ahead" button will execute the report. Once the report is executing, another report may be selected. The Quit button should be clicked when leaving the Report Interface window.

3.2 Input Parameters for Rereader reports

Before you run a report you need to make sure all the parameters needed to run a particular report are set to the desired values. There are five parameters; different subsets of these are required by specific reports (see Sections 3.3 through 3.8)

The meaning of each of the parameters shown in Figure 4 is:

Sentence Trace File: This is a file produced by running the CAPS ReReader simulation with the "Save sentence data" option chosen. The file name specified here is the same that you specified during the simulation. (This file is needed for Reports 6 and 7.)

Memory Trace File: This file is produced by running the CAPS ReReader simulation with the "Save memory data" option chosen. The file name should be the same one you specified for the simulation. (This file is needed for Reports 1 through 5.)

Report Output File: This is the file where the output of the report will be stored. You can choose any file name you want. Please try to be consistent - refer to Appendix B for more details about how to choose file names in the VAX. This output file is required for all Reports except Report 7. The report output files can be printed or transferred to PC or Macintosh computers for further analysis using application programs such as Microsoft Word or Excel.

Threshold: This is the activation threshold. This means that any CAPS working memory element whose activation is below this threshold will be considered inactive, and thus ignored in the calculations of a report. This parameter is used by Reports 2 through 5.

Cycle: This is the last cycle that you want to consider in the reports. The default value of zero will indicate that you want the report to use the very last cycle for which you have data in your data file. This parameter is used in Reports 1 through 5.

3.3 Report 1: Propositional Activation over all cycles (Table 3)

Cycles are listed in the columns and propositions in the rows. This report shows the activation in working memory of each proposition in each cycle. The number of the sentences in which propositions are first encountered (the one in which the proposition existence is established) are also listed in the leftmost column.

Output format: Excel formatted file.

Required parameters: Memory Trace File, Report Output file, and Cycle.

3.4 Report 2: Active Cycles and Re-instantiations (Table 4)

For all propositions encountered till the last specified cycle, this report provides the five columns of information.

- 1) The sentence number of the proposition listed in column 2.
- 2) The proposition number.
- 3) The number of times (i.e. cycles) in which the proposition remained active (i.e. over the specified threshold) in working memory.
- 4) The number of times the proposition was re-instantiated. We count as re-instantiations the number of times that the proposition reached the activation threshold after it had already become inactive, i.e. its activation had decayed enough to fall below the activation threshold. This is equivalent to the number of times a proposition reaches activation threshold excluding the first occurrence.
- 5) The activation of the proposition in working memory on the last cycle.

Output format: Excel formatted file.

Required parameters: Memory Trace File, Report Output file, Threshold, and Cycle.

3.5 Report 3: Summary of Long Term Memory Strengths (Table 5)

For each proposition encountered in the simulation run, or part of the run if so cycle specified by the user, this report provides four pieces of information:

- 1) The proposition number.

- 2) The long term memory strength of the proposition on the last specified cycle. The long term memory strength is calculated by adding the working memory activation of all active propositions (i.e., those over threshold) during all the preceding cycles.
- 3) The sum of the long term memory strengths of all the links that refer to the proposition listed in column 1. The links are considered to be bi-directional so that each link strength will be considered and added twice, once for each of the two propositions that the link refers to. This is provided in more detail in report 4. Each of the link strengths is calculated by adding the activation of links whose working memory activation is above threshold, over all the cycles preceding the specified cycle.
- 4) The sum of columns 2 and 3, which is another measure of the strength of the proposition.

Output format: Excel formatted file.

Required parameters: Memory Trace File, Report Output file, Threshold, and Cycle.

3.6 Report 4: Detailed Long Term Memory Strengths (Table 6)

This report details the information summarized in report 3. Instead of just providing the sum of all link strengths to a proposition, the link strengths of each individual link to a proposition are provided. The proposition and sentence number are also included in the report.

Output format: Text file.

Required parameters: Memory Trace File, Report Output file, Threshold, and Cycle.

3.7 Report 5: Detailed short term memory activations (Table 7)

This report provides the working memory activation of each active proposition and link in the specified cycle. Although the output format is similar to report 4, the meaning is different, since no sum of the activations takes place.

Output format: Text file.

Required parameters: Memory Trace File, Report Output file, Threshold, and Cycle.

3.8 Report 6: Reading Behavior Summary (Table 8)

This report indicates information about the number of times each sentence was read, skimmed, reread, initiated a reread, or ended a reread. Six columns of information are provided for that purpose:

1) Sentence. This is the sentence number.

2) Read. Indicates how many times the sentence was read (i.e. loaded with full activation and fully processed).

3) Skimmed. Indicates how many times the sentence was skimmed (i.e. loaded with reduced activation and not fully processed)

4) Reread. A reread happens when the same sentence is read again immediately after a previous read, with no reading or skimming of other sentence in the middle. This columns indicates how many times this happened with the sentence listed in column 1.

5) BackTo. This indicates how many times the sentence in column 1 was the end of a backtracking episode.

6) BackFrom. This indicates how many times the sentence in column 1 initiated a backtracking episode.

A graphic representation of this report is shown in Figure 5. Processing cycles are listed on the x axis while the sentence numbers are listed on the y axis. Sentences which are read in a particular cycle are shown by wide solid lines. Sentences which are skimmed during particular cycles are shown as narrow dashed lines.

Output format: Excel formatted file (Table 8) or Graphic Display (Figure 5)

Required parameters: Sentence Trace File, Report Output file.

4. Advanced Features

The foregoing sections of this manual provide the "basic operating procedures" for the ReReader simulation model. There are additional features that increase the flexibility of ReReader. We describe these features in this section.

4.1 Fixed Path Option

Under normal operation, the user specifies the Activation Cap and Motivation parameters and the "path" that the simulation takes is determined by the operation of the simulation. It is also possible to pre-specify a path that the simulation will use.

The ReReader Initialization Screen (Figure 3) contains two buttons: "Use Reading Strategies" and "Follow fixed path specified in Path file." These buttons allow the user to set the ReReader simulation mode. If you select the first mode (use reading strategies) ReReader will run normally, i.e., it will automatically choose what sentence to process next based on the results of the prior sentence and the parameters used to select reading strategies (see sections 5.4 - 5.7). If the user chooses the second mode (follow fixed path), ReReader will choose the next sentence to process according to the order specified by the Path File (see Appendix C.5).

In the path following mode, ReReader loads sentences into short memory and processes them in a predefined sequence. The processing of each individual sentence is identical to the normal mode (i.e. within-sentence links, between-sentence-links, activation boost, etc. are handled in the same manner), except for the fact that the measures of text coherence have no effect on the order in which sentences are processed. This mode is provided in order to help correlate the results of the simulation with human data. The path followed by human subjects can be retraced by ReReader. Simulation results can then be compared with actual behavioral data (for example, the simulated long term memory trace can be compared with actual recall data).

4.2 Saving Memory and Sentence Trace Files

Both the memory and the sentence trace files are normally saved when a simulation is run. Since this files are very long, they may end up filling up the disk. (Files that are not needed should be deleted.) It is possible to prevent the creation of either the memory or the sentence trace file if one or the other is of no interest to the user and space is a problem. The initialization screen (Figure 3) contains buttons that permit the user to specify save or do not save for the sentence data or the memory data. The memory trace file should be saved if you plan to run any of the other reports that summarize short or long-term memory data. The sentence trace file is needed to run the reading behavior reports (reports 6 and 7).

4.3 Parameters that can be changed with the User Interface

In addition to the Activation Cap and Motivation parameters there are several other parameters that can be manipulated. Some can be set through the ReReader User interface. Other parameters can only be set by modifying the ReReader source code, something which requires LISP programming expertise. The interface permits the user to set the initial activation given to the different sentence constituents, strategy link options, and coherence criteria options. The default values for constituent activations were described in section 1.2.1. Strategy link options allow the user to choose the set of linking rules that operate on a given run. The default is for all three to operate (argument overlap, propositional embedding and relational reference). At the same time, the default link activation value of 1 can be altered. The default for coherence criteria is for all five measures described above to be computed and used in the evaluation function. However, the user can deselect one or more of these indices of coherence.

4.3.1 Constituent Activations

When a sentence is loaded by ReReader, each of the elements of a sentence (i.e. the sentence *constituents*) get an initial activation. In our representation, a sentence is

described by one of several lines of words plus a set of propositions (See Section 2.1 or Appendix C.1). The initial activation of the words and propositions loaded each time ReReader processes a sentence can be set to different values if the given defaults are not adequate. Clicking on the button labeled "click here to change constituent activations" of Figure 3, will cause the window shown in Figure 6 to appear. The initial activation of concepts is set with the scale labeled "Word." The initial activation of propositions is set according to the proposition type (Exist, Modification, Relational Reference, and Verbal Predicate). For each of those types, the initial activation of the proposition can be set using the correspondingly labeled scale.

After setting the initial activation of the sentence constituents, clicking on the button labeled "Ready. Go Back" returns to the main interface window and activates the settings selected. "Ready, Go Back" must be pressed for the selected values to replace "old" values.

4.3.2 Link Strategy Options.

Links are created between propositions according to three rules: Argument Overlap, Propositional Embedding, and Relational Reference. Which of the rules is active can be specified. As well, the initial activation of the links created by each of these rules can be specified.

Clicking on the button on the ReReader Initialization Screen (Figure 3) labeled "Click here to change Link Strategy Options" causes the window shown in Figure 7 to appear. The menu allows the user to select which rules will be used to create links between propositions and what initial activations they will have. The user can also specify what additional activations will be added to propositions when the links are created (Propositional Boost).

After setting the Link Strategy Options, click on the button labeled "Ready. Go Back" to go back to the main interface window. The Link Strategy Options window will disappear. Until that is done, the selected settings will have no effect. Returning to the main menu before clicking on the "Ready, Go Back" button, will not update the values.

4.3.3 Coherence Criteria Options

Coherence criteria are the different parameters which are calculated by ReReader at the end of processing a sentence. These parameters are used in an evaluation function. The result of the evaluation function will determine which reading strategy should be used during subsequent processing in the normal ReReader mode.

Click on the button labeled "Click here to change Coherence Criteria Options" of Figure 3 and the window shown in Figure 8 will appear. Here you can select which of the possible coherence criteria should be in effect during the simulation (First Sentence of Passage, First sentence of paragraph, Previous sentence, Argument links present, Argument links missing). The default leaves all criteria *on*.

When you have the desired Coherence Criteria in effect, click on the button labeled "Ready. Go Back" to go back to the main interface window. Until you do that, the settings you selected will have no effect.

4.4 Parameters Modifiable through Changes to the Source Code

Several system variables are not accessible from the User Interface and can be modified only by directly changing the source code. This code is entirely contained in the text files INITIALIZE.L (Pyslisp code) and ReReader.L (CAPS productions and commands). All parameters described in this section can be found in the latter file.

1. Activation-Capping Scheme: CAPS ReReader supports three activation-capping schemes. Each was developed after flaws in a predecessor were discovered. ReReader, like most CAPS production systems, runs under the third one. This is specified early in the source file with the command (turn pre-spew-adjustment 3.0).

2. Activation Threshold: Working memory elements in CAPS have activations. For an element to match a production, it must match symbolically and have an activation greater than the specified threshold. When not explicitly stated in a production, this threshold defaults to 0.1. This can be changed by modifying the line (turn default-production-threshold 0.1), which occurs early in the source file.

3. Tracing: ReReader's path through a passage is reported at the sentence level. That is, every time a sentence is processed, several bits of information are displayed including the sentence being scanned, whether it's being read or skimmed, whether it was reached by moving forwards or backwards, the values of the measures of text coherence and the evaluation function, what course of action ReReader decided to take next based on the value of the evaluation function, the markers of problematic sentences that are still active (above the Activation Threshold described above), and the total number of CAPS cycles used to process the sentence and passage to that point. There are a number of internal cycles actually required to process a single sentence. The processing over these cycles in the form of production firing and activation constraints can be viewed in detail by changing selected parameters from *off* to *on* in the following command (found early in the source file): (turn trace-productions off trace-cycles off trace-all-spews off trace-cap off).

4. Starting Information: ReReader currently begins processing by reading forward from the first sentence of the passage. The initial task, reading or skimming, can be changed by altering the RHS action (<add> (current-task read)) in the first production, called *initialize-system*. The initial direction is set by the RHS action (<add> (current-direction forward)) in the same production. To begin processing at a sentence other than the first one, modify the RHS action (<add> (current-phase (start (<tok> sentence 1)))) in the next production, called *read-freely*. This modification has no effect when a fixed path is supplied because the first sentence on the path is the first sentence that will be read.

5. Phase Sequencing: CAPS is fully parallel, capable of firing all matched productions on each cycle. This allows several lines of processing to occur at the same time. ReReader currently processes sentences by cycling through several phases. The sequence in which these phases occur (or co-occur) is specified by working memory elements of the form (:ltm task direction phase-i :becomes phase-j) where the italics indicate variables. These elements are created in the production *read-freely* and *read-path* depending on whether ReReader is evaluating the coherence of the text after each

sentence and deciding what to do next based on the evaluation or reading a predetermined path through the passage. For instance, freely reading a sentence while moving forward requires cycling through six phases, one at a time, as indicated by the following working memory elements:

```
(:ltm read forward start :becomes scan-sentence)
(:ltm read forward scan-sentence :becomes link-within-sentence)
(:ltm read forward link-within-sentence :becomes link-between-sentences)
(:ltm read forward link-between-sentences :becomes fire-demons)
(:ltm read forward fire-demons :becomes evaluate-current-reading-strategy)
(:ltm read forward evaluate-current-reading-strategy :becomes end).
```

To change the "read forward" behavior so that scan-sentence, link-within-sentence, and link-between-sentences occur simultaneously, change the second and third elements above to:

```
(:ltm read forward start :becomes link-within-sentence)
(:ltm read forward start :becomes link-between-sentences).
```

To flip the order in which within-sentence and between-sentence links are created, alter the second, third, and fourth elements above to:

```
(:ltm read forward scan-sentence :becomes link-between-sentences)
(:ltm read forward link-between-sentences :becomes link-within-sentence)
(:ltm read forward link-within-sentence :becomes fire-demons)
```

There are two additional parameters, described below, that are not currently modifiable from the User Interface; however, we plan to provide access to them from the User Interface in a future version.

6. Number of Words to Scan: ReReader scans a sentence in one pass if it is skimming and two passes if it is reading. The first pass (which both modes share) activates an orthographic representation of each word (called a percept). The additional pass associated with reading is used to activate the propositions representing the semantic content of the sentence. Initially, it took one CAPS cycle to activate a percept and one to activate a proposition. Because propositions presumably require deeper

processing, and thus more time to activate, ReReader was changed to activate multiple percepts on each cycle. The RHS action (<number-of-words-to-scan> 3) in the production *initialize-system* sets this number. The default setting insures that percepts are activated three times as quickly as propositions.

7. Evaluation Function Threshold: The evaluation function returns a number based on the measures of text coherence. ReReader compares this number with the Evaluation Function Threshold and returns an evaluation (read on, mark problem, proceed backwards) depending on whether it is less than, equal to, or greater than the threshold. To set this threshold, alter the (<init-new-reading-strategy-threshold> 2) RHS action in the production *read-freely*.

The default values for these variables were determined based on procedures for tuning the operation of ReReader, following standard simulation techniques.

Appendix A: Login and Logout of the Rereader Account

Computer Environment

ReReader runs on a VAX workstation running the VMS operating system. The CAPS91 implementation of the CAPS software (developed at CMU) needs to be installed for ReReader to work. In order to run the ReReader interfaces, the X windows software needs to be running on the VAX workstation as well. It is also possible to run the interfaces on a Macintosh connected to the same network as the VAX workstation, provided the appropriate X terminal emulator software is running on the Mac. We have tested ReReader (using the Mac as display) with the MacX software.

A.1 Login to the ReReader Account in the VAX Workstation

Make sure that the previous user has logged out of the VAX station. You will then see a small login window where you can enter a username and a password. Type ReReader as the username and CAPS as the password. Sometimes there are some system messages overlapping the input window. If this happens, press the <Control> and <F2> keys simultaneously and you will get the login window.

To logout, find the "Session Manager" window and select the "Quit" item of the "Session" Menu. Some times the "Session Manager" window is not visible since it may have been covered by other windows or it may have been iconified (i.e. compressed into a little window labeled "ReReader"). To bring up a window that is covered, place the mouse on a portion of the screen not occupied by any window and click. You will get a menu with the options "Shuffle up" and "Shuffle Down". Select either one repeatedly until the desired window is placed on top of the others. If the window you want to use is iconified, just click on the corresponding icon and the window will open up.

Appendix B: Naming, Creating, and Transferring Files

B.1 File Names

Because ReReader runs on the VAX workstation, file names must conform to VAX file name conventions. A file name in the VAX system has two components: a file name proper, and a file extension. When new files are created, VAX will not automatically assign them the extension. Without the extension, the file can still be read by VAX but a lot of confusion may arise, mostly due to the fact that the user needs to maintain (and remember) a large number of files. The extensions help the user in keeping the files organized.

The file name proper may be of any length and may contain alphanumeric mixes. All letters are automatically converted into capitals by VAX. Two file names which differ only in the character case will be understood to have the same file name by the VAX, and the newer one will delete the older one. Therefore, care should be exercised in naming files.

The file extension is three characters long and normally indicates the type of file. The User should use the file extension consistently to differentiate the different types of files created by ReReader. For example, choose a standard file extension for all the report files created with Excel format. (We recommend the extension "XCL". Additional recommended extensions for each of the files used by ReReader are contained in Appendix C.) The file name proper and the file extension are separated by a period. A complete typical file name is: REPORT1-RESULT.XCL.

B.2 Creation of Files.

Files used by ReReader can be created with any editor or word processor that can store the file in plain text format. On a VAX, use the EDT editor, accessed by the "EDIT" command. (For more information, refer to the VAX manuals.) On a Macintosh, use a word processing package such as Word Perfect or Microsoft Word, and save the file in

"text" format. Then transfer the text file to the VAX. For example, a Word Perfect file should be saved using the Format option, "Text Export." To copy such a file from the Mac to the Vax, use software programs like telnet ftp or pacerlink. Make sure you copy the file to the ReReader account main directory. Once the file is copied, it can be accessed by the ReReader simulation. Remember to give the file a new name with the appropriate extension when you copy it.

B.3 Transfer of files from Macintosh to VAX using ftp

- 1) Run the telnet program on the Mac (by double clicking on the file named "NCSA telnet" or "Telnet ... MacTCP").
- 2) Select the item "Enable FTP" from the "file" menu.
- 3) Select the item "Open Connection" from the "file" menu. A dialog window will appear asking for a session name.
- 4) Type "capsvs.gpct.vanderbilt.edu" followed by a return. You will then get "Username: " prompt from the VAX.
- 5) Type ReReader as the username followed by a return. You will then get the "Password: " prompt from the VAX. Type "CAPS" followed by a return. You will then login to the VAX, and get the "\$" prompt.
- 6) Type the word "ftp" followed by a space (NO return).
- 7) Select the item "Send IP Number" from the "Network" menu.
- 8) Hit the return key TWICE.
- 9) Type "Get *filename*" followed by a return, where *filename* is the name of a file that resides in the currently selected Macintosh directory. This file will be transferred to the Vax.
- 10) Repeat step 8 for every file that needs to be transferred from the VAX to the Macintosh.
- 11) Type "bye" followed by a return to quit ftp.
- 12) Choose the item "Close Connection" from the "file" menu to leave the VAX system.

13) Choose the item "Quit" from the "file" menu to quit everything.

B.3 Transfer of files from VAX to Macintosh using ftp

First, follow the instructions 1) through 8) above.

9) Type "Put *filename*" followed by a return, where *filename* is the name of a VAX file, which will be transferred to the currently selected Macintosh directory.

10) Repeat step 8 for every file that needs to be transferred from the VAX to the Macintosh.

When you are done, follow instructions 11) through 13) above.

Appendix C: Simulation File Formats

C.0 File Names (passage, predicate, sentence trace, memory trace, path)

As in most other computer applications there are many files that need to be maintained in order to provide data to ReReader and save the results of the ReReader simulation. It is the responsibility of the user to create input files and choose adequate names for them. For example, the text and propositions of any particular passage have to be written and stored in a file using a format that ReReader can use. The name chosen for that file has to be entered through the ReReader interface so that the ReReader simulator is able to retrieve the correct passage in order to proceed with its simulation.

Default file names for the different files used in ReReader (both required and optional) are automatically provided when the ReReader interface is started (See Figure 5). The default names will rarely be the ones you need. To specify or modify a file name, move the mouse pointer to the square where the file name you want to change is located and click. Then modify the file name using delete and other keyboard keys.

The passage and the predicate files always need to be specified, since they contain the information specific to the particular passage to be simulated. Make sure the files exist and have been correctly edited. If you plan to save the sentence data or memory data, make sure that appropriate file names are provided. If you choose to simulate according to a fixed path, make sure the appropriate file exists and is correctly specified.

C.1 Passage File (Recommended extension: PSG)

The file should first contain the statement
(*convert-text* '(
followed by a series of sentence descriptions. Once all sentences of the passage have been described, indicate the end of the passage with two parentheses:
))

as shown at the end of Table C.1.

Sentence Descriptions

The description of each sentence consists first of the statement,
(*sentence sentence-number*)
where the *sentence-number* should start with 1 and be incremented for each new sentence.

If a sentence begins or ends a paragraph and the User's theoretical assumptions about text processing suggest that such a position would affect its salience to the reader, then ReReader must be told to add extra processing "weight" to that sentence.

- To "weight" a sentence that starts a paragraph, add the statement
(*start paragraph paragraph-number*)

- To "weight" a sentence that ends a paragraph, add the statement
(*end paragraph paragraph-number*)
where *paragraph-number* should start with 1 and be incremented for each new paragraph of the passage.

The next line or lines should be raw text, such as:

Distance is simply the space between two points.

Following each sentence is its propositional parse. Each proposition has a unique number. For example, (*5 distance*) indicates that proposition number 5 is the concept "distance." In addition to specifying concepts like "distance," propositions can define verbal relations among concepts. In this case, the concepts that are related are referred to in parentheses by their numbers rather than their names. For example, (*2 is^between (pp5 pp6)*) indicates that proposition number 2 is the relation "is between," which references proposition 5 (distance) and proposition 6 (points). In English, this reads "distance is between points."

Once all the propositions that come from a given sentence have been listed, one or more blank lines should be added before starting information pertaining to the next sentence.

C.2 Predicate File (Recommended extension: PRD)

The file format for the predicate file is shown in Table C.2. The predicate file contains information that a reader is assumed to possess before s/he starts reading a passage. That information will determine the initial activation given to different proposition types and also the manner in which different propositions are processed. In its current version, the only information ReReader can take into account is what category a given proposition falls into. We assigned propositions to four categories: *exist (a concept)*, *verbal*, *modification*, and *relational*. Each proposition used in the passage must be assigned to one of these four types and this must be specified in the predicate file.

To specify a proposition category, simply type (on a new line for each proposition) the following information:

(category proposition-name)

where *category* is one of the categories listed above, and *proposition-name* is simply the proposition itself, as it appeared in the passage file. For example,

(exist distance) indicates that the proposition "distance" is in the "exist" category.

At the beginning of each predicate file there should be the statement:

(incorporate-predicates '(

Once the passage propositions have been described, there should be two parentheses to indicate the end of the predicate file, i.e., *))*, placed as shown in Table C.2.

C.3 Memory File (Recommended extension: MEM)

The memory output file contains the simulated state of working memory after each sentence processing cycle. The beginning of such an output file is shown in Table C.3.

The purpose of this file is to store the results of a simulation run so that different types of reports can be produced from that simulation run. The output is not intended to be useful for direct inspection because the amount of data will be very large and disorganized.

For each cycle, the contents of short term memory are written out with a new group of information lines enclosed by parentheses. The useful information comes from two types of lines, as described below.

1) ((*sentence1 proposition1 is^between*) . 2.4)

This line shows that, during the current cycle, there is a proposition numbered 1, whose name is *is^between*, and which was first encountered in sentence number 1. The activation of this proposition at the end of the current cycle was 2.4.

2) ((*proposition1 :links-to proposition5*) . 1.0)

This line shows that at the end of the current cycle there is a link between proposition 1 and proposition 5 with an activation value of 1.0.

C.4. Sentence trace file (Recommended extension: SEN)

The sentence trace file provides a variety of information about the results of sentence simulations. Although some of the reports use this file as input, the information in this file is mainly intended as a raw data source. A portion of a sentence file is shown in Table C.4. The information provided includes the number of cycles to process the specific sentence and the cumulative number of cycles. Once sentence 2 is read, information is provided about the measures of text coherence and the outcome of the evaluation function. This information is provided in the *demons* line and the first 5 numbers in parentheses correspond to the text coherence measures specified in section 1.2.5. The sixth number is the motivation level. The strategy specifies the outcome of the evaluation function.

C.5. Path file (Recommended extension: PTH)

ReReader can operate in one of two modes. In one of them, the *path following* mode, ReReader simulates human data by reading the sentences of a passage in a pre-specified order rather than having the order determined by the on-going results of the simulation. The format of a pre-specified path is shown in the following:

1 5 15 10

This file specifies that sentence 1 should be read first, followed by sentences 5, then 15, and finally 10. The sentence numbers can be separated by spaces or can appear on new lines.

```

(convert-text '(
(sentence 1)
(start paragraph 1)
distance is simply the space between2 2 points
(1 is (pp4 pp5))
(2 is^between (pp5 pp6))
(3 quant (pp6 2))
(4 distance)
(5 space)
(6 points)

(sentence 2)
measured2 with a standard^unit such as the kilometer or mile
the result is called absolute^distance and is2 what most
persons probably think of when they think of distance
(20 measure (someone1 pp4 pp25))
(21 isa (pp25 pp26))
(22 isa (pp25 pp27))
(23 result (pp20 pp24))
(24 type^of (pp4 pp28))
(25 standard^unit)
(26 kilometer)
(27 mile)
(28 absolute^distance)
(29 think (pp32 pp31))
(30 mod (pp32 most))
(31 is^equal (pp28 pp4))
(32 persons)

(sentence 3)
(end paragraph 1)
however there is also something called relative^distance which
is when absolute^distance is influenced2 by other factors
(50 however (pp29 pp51))
(51 type^of (pp4 pp52))
(52 relative^distance)
(53 when (pp54 pp52))
(54 influence (pp56 pp28))
(55 mod (pp56 other))
(56 factors)

) )

```

Table C.1: Passage File

(incorporate-predicates '(

(exist distance)

(exist space)

(exist points)

(exist standard^unit)

(exist kilometer)

(exist mile)

(exist absolute^distance)

(exist persons)

(exist relative^distance)

(exist factors)

(modification quant)

(modification mod)

(modification type^of)

(relational however)

(relational influence)

(relational when)

(verbal is)

(verbal isa)

(verbal is^between)

(verbal measure)

(verbal result)

(verbal think)

(verbal is^equal)

))

Table C.2: Predicate File

```
(
((sentence1 proposition2 is^between) . 2.4)
((sentence1 proposition1 is) . 2.3)
((sentence1 proposition3 quant) . 1.2)
((sentence1 proposition5 space) . 1.2)
((sentence1 proposition6 points) . 1.2)
((sentence1 proposition4 distance) . 1.1)
((proposition1 :links-to proposition4) . 1.0)
((proposition1 :links-to proposition5) . 1.0)
((proposition2 :links-to proposition5) . 1.0)
((proposition2 :links-to proposition6) . 1.0)
((proposition3 :links-to proposition6) . 1.0)
((proposition1 :links-to proposition2) . 1.0)
((proposition2 :links-to proposition3) . 1.0)
)
```

```
(
((sentence2 proposition29 think) . 1.014046993867223)
((sentence2 proposition23 result) . 0.8527096855242637)
((sentence2 proposition20 measure) . 1.13633076166654)
((sentence2 proposition31 is^equal) . 1.284952913829636)
((sentence1 proposition2 is^between) . 0.7230155303685141)
((sentence2 proposition21 isa) . 0.8951659867222199)
((sentence2 proposition22 isa) . 0.9230288953991904)
((sentence1 proposition1 is) . 0.9458789808527028)
((sentence2 proposition24 type^of) . 0.7590274545116866)
((sentence1 proposition3 quant) . 0.3615077651842571)
((sentence2 proposition30 mod) . 0.5491496964887308)
((sentence2 proposition25 standard^unit) . 0.548555234400117)
((sentence1 proposition5 space) . 0.3615077651842571)
((sentence1 proposition6 points) . 0.3615077651842571)
((sentence2 proposition32 persons) . 0.5765007626333879)
((sentence2 proposition27 mile) . 0.4654028548960273)
((sentence2 proposition26 kilometer) . 0.4572508051421736)
((sentence1 proposition4 distance) . 0.5843712156684457)
((sentence2 proposition28 absolute^distance) . 0.5233662630944462)
((proposition1 :links-to proposition4) . 0.3012564709868809)
((proposition1 :links-to proposition5) . 0.3012564709868809)
((proposition2 :links-to proposition5) . 0.3012564709868809)
((proposition2 :links-to proposition6) . 0.3012564709868809)
((proposition3 :links-to proposition6) . 0.3012564709868809)
((proposition1 :links-to proposition2) . 0.3012564709868809)
((proposition2 :links-to proposition3) . 0.3012564709868809)
((proposition20 :links-to proposition25) . 0.7954758559370419)
((proposition21 :links-to proposition25) . 0.7954758559370419)
((proposition22 :links-to proposition25) . 0.7954758559370419)
((proposition1 :links-to proposition20) . 0.9481965236465676)
)
```

Table C.3: Partial Printout of a Memory File

read sentence1

cycle count on sentence1: 27
total cycles thus far: 27

read sentence2

demons: (6 0 0 15 0 1.0) : 7.0
strategy: read-on
continue reading forward normally

cycle count on sentence2: 48
total cycles thus far: 75

read sentence3

demons: (2 0 9 9 0 1.0) : 4.0
strategy: read-on
continue reading forward normally

cycle count on sentence3: 37
total cycles thus far: 112

read sentence4

demons: (2 0 0 4 0 1.0) : 3.0
strategy: read-on
continue reading forward normally

cycle count on sentence4: 29
total cycles thus far: 141

Table C.4: Begining of Sentence File

Appendix D: MOTIF Interface Components.

Window Components: Buttons, Text Windows, and Scales

The interface software used for ReReader is based on a software package called MOTIF. This package uses some interface conventions (i.e, ways to manipulate the mouse, windows, etc.) that are quite similar to, but not identical with, the familiar Macintosh interface conventions.

(General note: In the following descriptions, *to click* means to push the (only) mouse button on a Macintosh mouse, or to push the leftmost button on a VAX mouse. The other buttons on the VAX mouse are not used in ReReader. To *double click* means to click twice in rapid succession on that same button.)

D.1 Buttons

There are three different kinds of buttons, reflecting three types of choices.

1. Pushbutton: A pushbutton is a window surrounded by a border that has an associated text label. When you click inside the borders of a pushbutton, then some action (normally explained by the label) is taken. For example, when you click on the pushbutton labeled "Quit" in the button of Figure 3, ReReader will quit, the interface window will disappear, and the system will return to the VAX prompt (the "\$" sign).

2. Toggle Button: A toggle button is used for options that can be turned on or off, like a light switch. The button is *on* when the square next to the option label is dark (looking like a button which has been pushed), and *off* when the square is the same color as the rest of the window. To change the state of a toggle button (from on to off and vice-versa), click on top of the button square or its label. Although toggle buttons sometimes are grouped together for convenience's sake, they can each be toggled independently.

3. Radio Button: A radio button is used to select an option out of a possible set of options, like a radio station tuner button. The button is on when the diamond next to the option label is dark, and off when the diamond is the same color as the rest of the

window. To change the state of a radio button (from on to off and vice-versa) just click on top of the button diamond or its label. Radio buttons come in groups and only one of the radio buttons of a group can be on at any particular time. If a different radio button is turned on, all other radio buttons will automatically be turned off.

4. Text Window: A text window allows you to enter and/or modify text parameters. The text parameters used in ReReader are the different file names that specify which passage file to use, in which file to store the output of a report, etc. The text window is actually a small editor. Although many fancy editing operations are possible (they are explained in some of the voluminous X window system references), you can rely upon simple editing operations that you already know for the purposes of using ReReader. For example, the Delete key deletes backwards, clicking in the middle of the text allows you to start inserting text in that position, and so forth.

No action is taken when the text in a text window is modified. The text in the window will only be used after some action is indicated by clicking on a pushbutton. For example, if you enter the name of a report output file in a text window, the name is actually retrieved and used by the program once you push the button "Go Ahead and Run Report"

5. Scale: A scale is used to set a numeric parameter. For example, you can set the activation cap that ReReader works with anywhere along a range from 0 to an upper limit of, say, 200. The values are set by moving a small rectangular pointer, called a *notch*, with the mouse. The possible range is indicated by the scale itself.

Notches can be moved in two ways: dragging or incremental clicking. The User can drag the notch to the desired location. (*To drag* means to move an object on the screen by positioning the mouse pointer on an object, pressing down the button without releasing it, moving the mouse pointer while holding down the mouse button until the object is at its desired location, and then releasing the button.) It is easy to set the scale to the maximum or minimum values by dragging the notch all the way to the right or left of the scale.

If there are many possible scale values, then it may be difficult to drag the notch to the desired setting. Incremental clicking may be more precise. The User positions the mouse pointer on the scale to the right or the left of the notch in the direction that you want the notch to move. Each click, moves the notch a large and exact incremental value. The User can also position the notch all the way to the left/right of the scale by positioning the pointer to the left/right of the notch, pressing on the mouse button, and holding the button down until the notch reaches the left/right end of the scale.

References

- Goldman, S. R. & Saul, E. U. (1990). Flexibility in text processing: A strategy competition model. *Learning and Individual Differences*, 2, 181-219.
- Just, M.A., & Carpenter, P. A. (1992). A capacity theory of comprehension: Individual differences in working memory. *Psychological Review*, 99, 122-149.
- Kintsch, W. (1988). The role of knowledge in discourse comprehension: A construction-integration model. *Psychological Review*, 95, 163-182.
- Turner, A. & Greene, E. (1978). The construction and use of a propositional text base. *JSAS Catalog of Selected Documents in Psychology*, 8, 58.

Sample Propositional Parse for the First Sentence of the Distance Passage

Sentence: Distance is simply the space between two points.

Proposition	Type of Proposition	Activation
p0 simply(p1)	modification	1
p1 Is(p4,p5)	verbal predicate	2
p2 is^between(p5,p6)	modification	1
p3 quant(p6,2)	modification	1
p4 distance	concept	1
p5 space	concept	1
p6 points	concept	1

Table 1: Propositional representation of a sentence

Distance is simply the space between two points. Measured with a standard unit such as the kilometer or mile, the result is called absolute distance and is what most persons probably think of when they think of distance. However, there is also something called relative distance, which is when absolute distance is influenced by other factors.

There are a number of factors that affect distance measurement. Economic distance is measured in relation to the cost of movement from one place to another. There is a cost involved in any movement, in terms of either money or energy. The cost of shipping something by water is usually about one-tenth the cost over land, despite the fact that land routes are usually shorter.

Distance can be measured relative to time. Maps that use travel time instead of mile markers as the units of measure distort the usual spatial relations among locations. Travel time from a single point, such as the central business district, to a location 10 miles north may be the same as travel time to a location 30 miles south.

Psychological distance is measured relative to our perceptions, which may vary. What may seem like a long trip to some individuals may seem short to others. Even the same route going and coming can seem different to a single traveler, depending on road conditions, time of day, or anticipation of the end of the trip.

Distance can be measured relative to direction of movement. When 2 points are located at different elevations, movement from point A to point B may not be as easy as from B to A. The mileage between the points may be equal but uphill movement is harder than downhill.

Distances in geographic space can seem longer due to friction with obstacles. For example, large crowds or heavy traffic are considered friction because they slow down our movement. Often we are willing to travel farther in order to reduce friction, such as when we go to a suburban mall rather than to a downtown store to avoid traffic and crowds.

Table 2: Distance passage

STM activation of propositions after each cycle

Sent.	Prop.	Cycles		
		1	2	3
1	1	2.3000	0.9459	0.4206
1	2	2.4000	0.7230	0.2650
1	3	1.2000	0.3615	0.1325
1	4	1.1000	0.5844	0.2882
1	5	1.2000	0.3615	0.1325
1	6	1.2000	0.3615	0.1325
2	20	0.0000	1.1363	0.4904
2	21	0.0000	0.8952	0.3281
2	22	0.0000	0.9230	0.3383
2	23	0.0000	0.8527	0.3125
2	24	0.0000	0.7590	0.4262
2	25	0.0000	0.5486	0.2010
2	26	0.0000	0.4573	0.1676
2	27	0.0000	0.4654	0.1706
2	28	0.0000	0.5234	0.3398
2	29	0.0000	1.0140	0.5196
2	30	0.0000	0.5491	0.2012
2	31	0.0000	1.2850	0.6189
2	32	0.0000	0.5765	0.2113
3	50	0.0000	0.0000	0.6357
3	51	0.0000	0.0000	1.1336
3	52	0.0000	0.0000	0.6998
3	53	0.0000	0.0000	0.7103
3	54	0.0000	0.0000	1.0019
3	55	0.0000	0.0000	0.6729
3	56	0.0000	0.0000	0.7580

Table 3: Report 1 Output

Active cycles and Re-instantiations

Sent	Prop	Active	Re-inst	Last act.
1	1	3	0	0.4206
1	2	3	0	0.2650
1	3	3	0	0.1325
1	4	3	0	0.2882
1	5	3	0	0.1325
1	6	3	0	0.1325
2	20	2	0	0.4904
2	21	2	0	0.3281
2	22	2	0	0.3383
2	23	2	0	0.3125
2	24	2	0	0.4262
2	25	2	0	0.2010
2	26	2	0	0.1676
2	27	2	0	0.1706
2	28	2	0	0.3398
2	29	2	0	0.5196
2	30	2	0	0.2012
2	31	2	0	0.6189
2	32	2	0	0.2113
3	50	1	0	0.6357
3	51	1	0	1.1336
3	52	1	0	0.6998
3	53	1	0	0.7103
3	54	1	0	1.0019
3	55	1	0	0.6729
3	56	1	0	0.7580

Table 4: Report 2 Output

LTM Strengths of propositions at end of processing

Prop. Number	Self Str.	Link Str.	Total Strength
1	3.6665	9.0430	12.7095
2	3.3880	5.6466	9.0346
3	1.6940	2.8233	4.5173
4	1.9725	6.2197	8.1922
5	1.6940	2.8233	4.5173
6	1.6940	2.8233	4.5173
20	1.6268	10.0343	11.6611
21	1.2232	4.3480	5.5712
22	1.2613	4.3480	5.6093
23	1.1652	2.1740	3.3392
24	1.1852	8.7813	9.9665
25	0.7496	3.2610	4.0106
26	0.6248	1.0870	1.7118
27	0.6360	1.0870	1.7230
28	0.8632	4.0160	4.8792
29	1.5337	5.1030	6.6367
30	0.7504	2.1740	2.9244
31	1.9039	8.7813	10.6852
32	0.7878	2.1740	2.9618
50	0.6357	1.6258	2.2615
51	1.1336	7.6403	8.7739
52	0.6998	2.1143	2.8141
53	0.7103	2.1143	2.8246
54	1.0019	5.5821	6.5840
55	0.6729	1.4096	2.0825
56	0.7580	2.3305	3.0886

Table 5: Report 3 output

LTM strengths of propositions and links

Sent. Proposition

1 prop1 is

Self Strength: 3.6665

Link Strengths to:

prop2 1.4117

prop4 1.4117

prop5 1.4117

prop20 1.2957

prop24 1.2957

prop31 1.2957

prop51 0.9210

Total Link Strength: 9.0430

Total Strength of proposition: 12.7095

1 prop2 is^between

Self Strength: 3.3880

Link Strengths to:

prop1 1.4117

prop3 1.4117

prop5 1.4117

prop6 1.4117

Total Link Strength: 5.6466

Total Strength of proposition: 9.0346

Table 6: Report 4 output

STM activation of propositions and links at last cycle

Sent. Proposition

1 prop1 is

Self Activation: 0.4206

Link Activation to:

prop2 0.1104

prop4 0.1104

prop5 0.1104

prop20 0.3475

prop24 0.3475

prop31 0.3475

prop51 0.9210

Total Link Activation: 2.2947

Total Activation of proposition: 2.7153

1 prop2 is^between

Self Activation: 0.2650

Link Activation to:

prop1 0.1104

prop3 0.1104

prop5 0.1104

prop6 0.1104

Total Link Activation: 0.4416

Total Activation of proposition: 0.7066

Table 7: Report 5 output

Sentence Reading Behavior Summary

Sent.	Read	Skim	Reread	BackTo	BackFrom
1	1	0	0	0	0
2	1	0	0	0	0
3	1	0	0	0	0
4	1	0	0	0	0
5	1	0	0	0	0
6	1	0	0	0	0
7	1	0	0	0	0
8	1	0	0	0	0
9	2	0	0	1	0
10	2	1	0	0	0
11	2	1	0	0	0
12	2	1	0	0	0
13	2	1	0	0	0
14	1	1	0	0	1
15	1	0	0	0	0
16	1	0	0	0	0
17	1	0	0	0	0
18	2	0	0	1	0
19	1	1	0	0	1

Table 8: Report 6 output

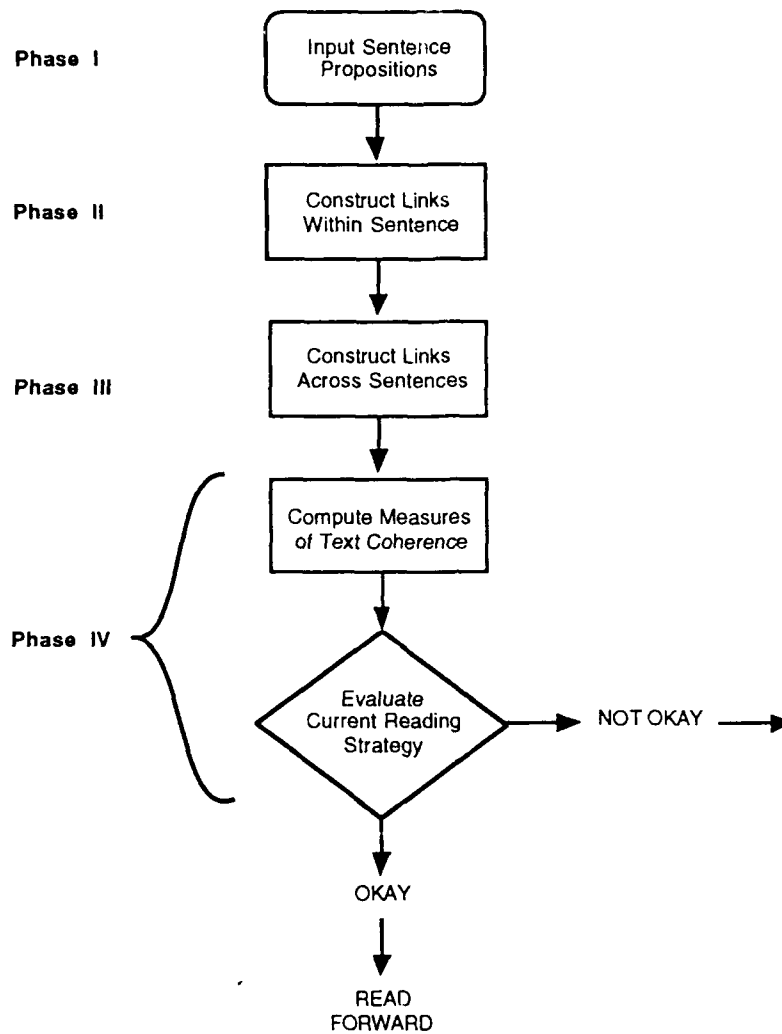


Figure 1: Phases of The ReReader Simulation

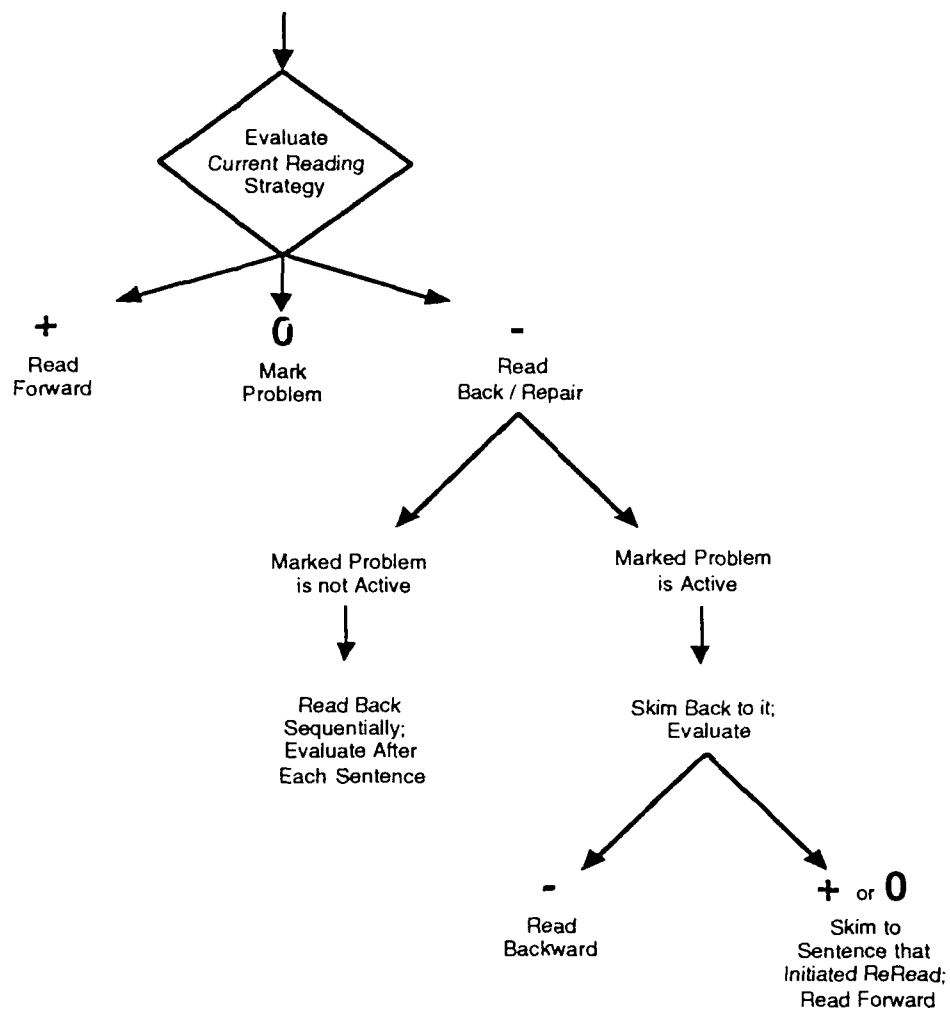


Figure 2: Selection of Reading Strategies

REREADER INTERFACE

CAPS REREADER Initialization

Passage File:

Predicate File:

☐ Don't save sentence data

☒ Save sentence data in the sentence trace file

Sentence Trace File:

☐ Don't save memory data

☒ Save memory data in the memory trace file

Memory Trace File:

☐ Use reading strategies

☒ Follow fixed path specified in Path file

Path File:

50 1

Activation Cap Motivation

[Click here to change Constituent Activations](#)

[Click here to change Link Strategy Options](#)

[Click here to change Coherence Criteria Options](#)

[Click here for HELP](#) [Quit](#) [Go on and Simulate](#)

Figure 3: Main Window for ReReader Simulator

REREADER Report Selection

- ☒ Proposition activation over all cycles (Excell format)
- ☐ Active-cycles and re-instantiations (Excell format)
- ☐ Summary Long Term Memory Strengths (Excell format)
- ☐ Detailed Long Term Memory Strengths
- ☐ Detailed Short Term Memory Activations
- ☐ Reading Behavior Summary (Excell format)
- ☐ Reading Behavior Graph (Displayed)

Sentence Trace File:

Memory Trace File:

Report Output File:

.100

Threshold

0

Cycle (Use 0 for last cycle)

Figure 4: Report Interface Menu

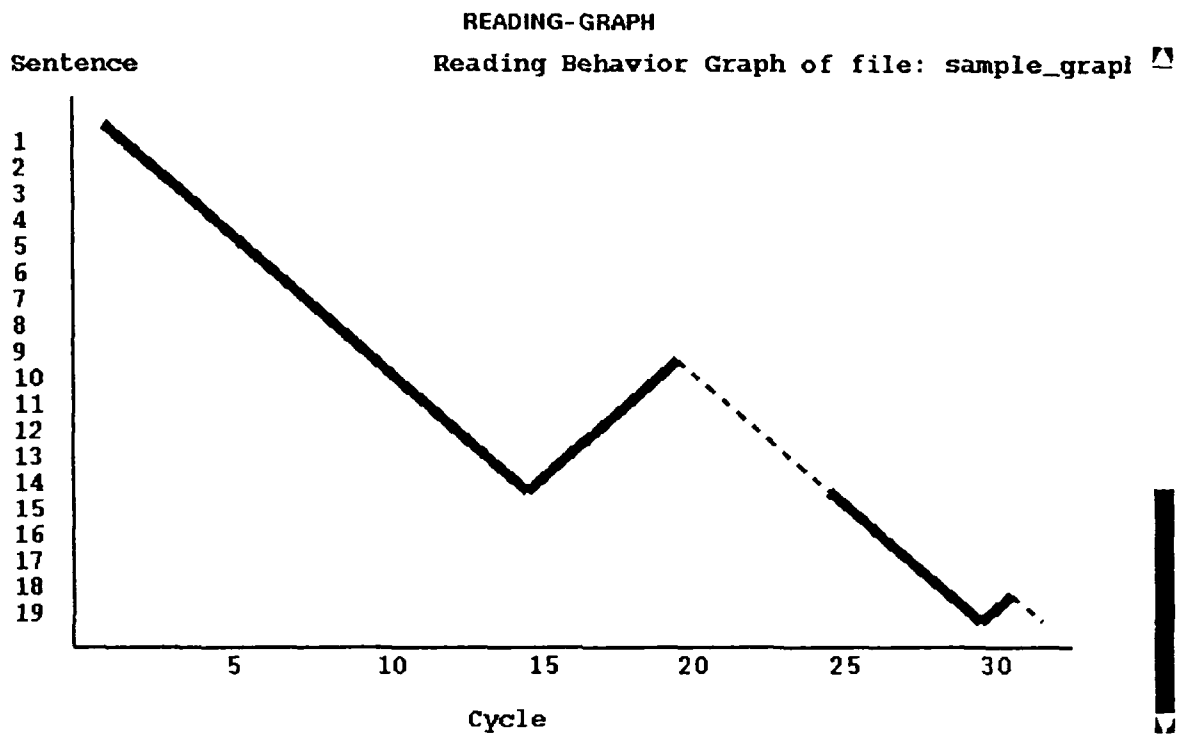


Figure 5: Graphic Display of Report 6

Link Strategies

1.00

Word

1.00

Exist

1.00

Modification

1.00

Relational Reference

2.00

Verbal Predicate

Ready. Go Back Need Help

Figure 6: Constituent Activations Menu

Link Strategies

☒ Argument Overlap

1.00

☒ Propositional Embedding

1.00

☒ Relational Reference

1.00

Activation Boost

.10

Ready. Go Back Need Help

Figure 7: Link Strategy Options

Link Strategies

- ☐ First Sentence of Passage
- ☐ First Sentence of Paragraph
- ☐ Previous Sentence
- ☐ Argument Links Present
- ☐ Argument Links Missing

Figure 8: Coherence Criteria Options