STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

18 February 1969

Semiannual Technical Letter Report 1
Covering the Period 9 August 1968 through 8 February 1969
Stanford Research Institute Project 7079

STUDY FOR THE DEVELOPMENT OF HUMAN INTELLECT AUGMENTATION TECHNIQUES

by

D. C. Engelbart

and Staff of the Augmented Human
Intellect Research Center

Contract NAS1-7897

Prepared for

National Aeronautics and Space Administration
Langley Research Center, Langley Station
Langley, Virginia 23365, Mail Stop 126

Copy No. 96

ABSTRACT

This report covers progress made during six months of a continuing research program in the Augmented Human Intellect Research Center (AHIRC) of Stanford Research Institute.

The program is directed toward the discovery of principles and techniques for the augmentation of human effectiveness in intellectual tasks by means of direct, on-line computer aids used on a full-time basis.

A considerable portion of the report is devoted to developments in the "On-Line System" (NLS), which is a unified system incorporating many of the computer aids that have been developed in the project.

CONTENTS

ABSTRACT i

FOREWORD iii

I GENERAL 1

II NEW NLS FEATURES 3

III TODAS 9

IV GODOS 10

V NIC 11

VI SERVICE-SYSTEM DEVELOPMENTS 17

VII FUTURE PLANS 19

Appendix A: AHI PARTICIPATION AT THE FJCC 24

Appendix B: THE NLS VECTOR PACKAGE 31

Appendix C: THE INFORMATION-RETRIEVAL SYSTEM 35

Appendix D: THE NLS CONTENT ANALYZER 40

Appendix E: THE NLS LINKING FEATURE 52

Appendix F: TODAS USER'S GUIDE 56

Appendix G: GODOS USER'S GUIDE 60

Appendix H: NIC SYSTEM SPECIFICATIONS (PART I) 66

Appendix I: PLAN FOR REWRITING COMPILERS 70

Appendix J: NETWORK DECODE-ENCODE LANGUAGE 78

Appendix K: IMPLEMENTATION PLAN FOR THE NIC 82

SIGNATURE SHEET 90

FOREWORD

The Augmented Human Intellect Research Center (AHIRC) operates under multiple sponsorship by NASA, ARPA, and the U. S. Air Force. Although this report applies specifically to NASA Contract NAS1-7879, not all of the work reported was funded exclusively by NASA. The Center's multiple funding is used in a highly integrated, flexible manner. For this reason, explicit separation of funding areas for a report such as this one would be difficult and would result in an unnatural division of material from a technical-information point of view.

AHIRC is a research center operating within Stanford Research Institute. It is devoted to research on techniques and principles for the augmentation of human intellectual processes by means of direct computer aid to intellectual (information-handling) tasks.

This report was composed, organized, formatted, and typed entirely by means of some of the computer aids developed by AHIRC. The main body of the report is a general account of activities in AHIRC during the last six months and of plans for the future. Several of the areas treated in the main body are expanded in greater detail in the appendices.

I GENERAL

The three most important aspects of our progress during this six-month period were as follows:

Becoming operational with our six-station On-Line System (NLS) facility

Participating in the Fall Joint Computer Conference, including the preparation for the presentation (see Appendix A for a full account of this participation)

Making a significant step ahead in our "bootstrapping" by integrating a significant portion of our software activity with NLS.

We also have implemented quite a few new user features into NLS, and have made significant progress in planning and specifying services to be developed for the Network Information Center.

Having gotten our time-shared, multiconsole system operational, we have learned that at full-usage rate the system will only support six or seven work stations -- more than this degrades the response for all users to an unacceptable level.

We also realized that the evolution and maintenance of the service system was costing considerably more than we had budgeted, causing the research activities to be resource-starved.

A series of negotiations with ARPA during this past six months has led to an increase in the level of funding (through our RADC contract) to bolster service-system support, and especially to provide some additional core memory. The additional memory capacity will alleviate the time-sharing bottlenecks so that all 12 work stations may be served adequately.

We have also submitted two other facility-expansion proposals to ARPA:

One proposal is to develop a multiprocessor subsystem that would take much of the higher-frequency service load off the 940 and enable us to support about 32 NLS users (some over the network, but 24 here at SRI -- the proposal also includes adding 12 more work stations).

The other proposal is to provide an on-line CRT-to-film output capability such that we can automatically publish multipage documents (or microfiche) containing the mixture of text and illustrations that we are now beginning to accommodate in NLS (this facility is especially relevant to the proposed services of the Network Information Center).

In the miscellaneous category, there have been several noteworthy developments:

As a byproduct of our FJCC activity, we have developed a cheap and flexible technique for making 16-mm movies directly from the display screen of a slaved console, with sound from a microphone at the command console.

Using this technique, we now have a 1-hour, 40-minute film that captures the projected video and audio that the audience of our FJCC session saw and heard. We also have a 22-minute reel specially made for the subsequent ARPA contractors' meeting in Hawaii. The big movie is an overview of our project; the smaller movie is a special feature oriented toward programmers.

We plan to keep copies of both movies available for loan to interested parties.

We have evolved a prototype work station that departs significantly from conventional console design. In association with this, we are experimenting also with work-space layout and equipment (walls, tables, shelves, etc.) of a new degree of flexibility and modularity.

The work station uses the same display and control devices, but houses and mounts them differently. The controls are separated from the display and are integrated into a self-contained "control console" that may be mounted conveniently on the users chair or placed upon any flat surface; the display is independently adjustable in position and attitude.

These developments have been made in cooperation with the Herman Miller Research Corporation, of Ann Arbor, Michigan. Their parent company has just come on the market with the new line of office furnishings (including the walls), and they have cooperated eagerly in styling and producing the prototype work station.

We expect within the next six months to make considerable progress toward housing our crew in experimental working space and equipping them with the new work stations.

With the increased level of our service-system activity, and with the development of NIC services becoming a major activity, we anticipate expanding our staff by perhaps a dozen people within the next few months.

II NEW NLS FEATURES

A. VECTOR PACKAGE

1. INTRODUCTION

The Vector Package is a first-stage graphics system operating as an integral part of NLS.

Every drawing created with the vector package is associated with a statement in an NLS file. The drawing consists of straight lines (vectors) and labels (text, with the same character set as NLS itself).

2. VECTORS

The Insert Vector command allows the user to define two endpoints with the mouse; a vector is then automatically drawn between these endpoints. The second endpoint may be used as the first endpoint of the next vector, or the next vector may start elsewhere.

After a vector has been created with this command, it may be moved, translated, deleted, or projected to the vertical or horizontal. Each of these operations makes use of a selection made with the mouse. The user points to the vector and hits the Command Accept (CA) button; the system then marks the ends of the vector with an O and an X.

In the Delete Vector command, the vector is immediately deleted upon execution of the command.

In the Move Vector command, the end marked X moves to a new position selected with the mouse, and the end marked O remains fixed. In the Translate Vector command, the end marked X aain moves to a new position selected with the mouse, but the end marked O also moves in such a way as to preserve the length and direction of the vector.

In the Horizontal and Vertical commands, the end marked X moves vertically until the vector is horizontal, or horizontally until the vector is vertical.

3. LABELS

In the Insert Label command, the user types the text of a label and this text is then "attached" to the cursor. The user moves it to the desired location and hits CA; the label is then fixed. In the Delete Label command, the user simply points to a label and it is deleted on CA; to move a label, the user may cause it to be "attached" to the cursor again and move it to any desired position.

4. OTHER COMMANDS

The user may move the entire drawing (set of existing vectors and labels) relative to the associated statement text, or he may delete the entire drawing and start over. A grid of dots may be superimposed for use as a guide in drawing vectors and locating labels.

A portion of the NLS User's Guide relating to the vector package is given as Appendix B.

B. KEYWORD RETRIEVAL SYSTEM

The keyword information-retrieval system is essentially a method of reordering a keyworded catalog of NLS files in accordance with some selected and weighted subset of the available keywords.

A catalog file for use with this system consists of two parts:

A list of keywords used, with information as to their meanings

A list of files referenced, with information as to their contents, machine-readable lists of applicable keywords, and machine-executable "link" text for retrieving the actual files.

To use the catalog, the user examines the keyword listing and selects (with the mouse) any keywords that relate to desired types of information. He may also assign weights to the keywords, on a scale from 1 to 10.

Upon execution, the system assigns "scores" to the entries in the file listing, according to the number and weights of keywords applying to each entry.

It then displays a list of entries with nonzero scores, in order of decreasing score. This constitutes an ordered list of files containing the types of information desired by the user.

The user may now retrieve individual files with the Jump to Link feature (described in another section of this report).

Detailed information on the operation of the information-retrieval system is given in Appendix C.

C. CONTENT ANALYZER

The Content Analyzer feature of NLS is a highly flexible system allowing the user to specify a particular "pattern" of content, and then cause display of only those statements in a file which satisfy that pattern.

The pattern specification is written into the file itself, as text, in a special high-level language. The pattern may be simple -- e.g., it may call for the occurrence anywhere in a statement of some particular word -- or it may be highly complex -- e.g., it may involve the occurrence of several words or characters in some special relationship within the statement.

The specification of simple patterns is simple and easy; complex patterns may require fairly intricate formulation.

Stored with every statement in a file is information giving the initials of the last person to change the text of the statement and the date and time on which this occurred. This information may be interrogated by the content analyzer: thus the user may write a pattern whose meaning is "display statements written or changed by anyone except me since the beginning of this week."

Detailed information on the operation of the content analyzer is given in Appendix D.

D. FILE CLEANUP

Every NLS file contains not only the actual text and drawings written into it but a great deal of information used by the system in transferring it from one location to another, relating statements to one another structurally, displaying the file, etc.

Occasionally, hardware errors cause bad spots in these data or bad characters in the text; also, because of the complexity of the software and its constant state of revision, software bugs sometimes cause errors in file information.

Such errors can be very serious in their effect on a file. Certain types make the file completely unoeadable by NLS, and these must be corrected by accessing the file directly in core with an on-line debugging system. Many errors, however, affect the user's use of the file only when a particular statement is displayed or a particular operation attempted.

A user command in NLS executes a program which can automatically correct many of these errors and can detect and list others. This system, called File Cleanup, has drastically reduced the number of

files lost through system errors, particularly those errors which are not immediately apparent but which may propagate into other, more serious errors when some operation is executed on the file.

E. LINK JUMPS

By means of "links," it is now possible for the user to move about in several different files as easily as he formerly moved in a single file.

All files used in this fashion must be stored on one of the rapid-access devices (disc or drum). The current configuration of the system limits the number of files that can be so stored, but plans for the near future include the ability to store many files in this fashion.

A link is a string of text specifying a particular file, a location in that file, and a set of viewing parameters to be used in displaying the file initially. The format of the link is (USERNAME,FILENAME,LOCATION:VIEWSPECS).

The user name is omitted if the referenced file belongs to the same user as the current file.

The file name is omitted if the link refers to a location in the current file.

If the location is omitted, the beginning of the file is assumed.

If the VIEWSPECs are omitted, the current VIEWSPECs are assumed.

This text string, which is readable by the user or by the system, is inserted in the file text in the same way that a cross-reference would be used in hardcopy. Note that it may refer to another file or to another location in the current file.

When a user encounters the link, he may follow it simply by giving the command "Jump to Link" and pointing to the link with the cursor. The referenced material is immediately displayed; the user may then return to the previous display, or he may follow another link if there is one.

Detailed information on the use of links is given in Appendix E.

F. VIEWCHANGE

A large set of user commands has been implemented to give the NLS

user the power to reformat his display. The various feedback entities normally displayed at the top of the screen may be individually moved, modified, or deleted, as may the main text area of the display.

Any of these display entities may be suppressed completely, or moved to a different part of the screen, or displayed in larger or smaller characters than normal. They may also be displayed in italic, underlined, or flashing characters.

The command feedback line may also be displayed in abbreviated form, showing only the initial letters of the words normally displayed.

In the main text area, the line length and the number of lines may be adjusted.

The two characters used for displaying the cursor spot ("bug") may be changed.

Once the user is satisfied with a new display format, he may cause the various display parameters that he has set to be stored as part of his file. At any future time he may then retrieve them with a user command and put them into effect.

One purpose of this system is to permit reformatting of the display for use in video communication. Thus a user may clear some area of his screen (e.g. top or bottom half, right or left half, or one corner) to be used for superimposing a video image of his own face, another user's face, another user's display, or some other image.

Experimentation with these techniques is still at an early stage, but some of the possibilities were explored quite extensively in the presentation for the 1968 Fall Joint Computer Conference, where a large screen with the speaker's console display projected on it was used as a primary means of communication to the audience.

G. PRINTER GRAPHICS

NLS files may be output on a Potter line printer, and this is the normal means of obtaining a quick hardcopy of a file for internal use, where high physical quality is not an objective.

Changes in the printer portion of the output processing system permit drawings created in NLS files with the vector package to be approximated on the printer by plotting lines as series of periods, vertical-bar characters, etc. The quality of these

"drawings" is usually good enough for intelligibility and for examination of page format.

III TYPEWRITER-ORIENTED DOCUMENTATION AID SYSTEM (TODAS)

The first stage of TODAS has been implemented and is in use. This system permits a trained typist to enter documents into the system via paper tape produced on a Teletype or other typwriter-like device.

> Documents entered in this fashion incorporate the NLS structured-statement format; this may be as elaborate as desired by an author or, in the case of a document not written with NLS in mind, may simply use paragraphs as statements in a one-level list.

The paper tape is read in via a simple process and creates a sequential file resembling files produced with QED. This file may then be converted to an NLS file by use of the NLS command Insert QED Branch.

The typist must create statement numbers if they are not present in the original copy, and must use a special character to indicate capitalization. Apart from this, the typist simply transcribes the material literally.

> The typist may correct simple typing errors by typing special control characters to cause deletion of words just typed or of a statement being typed. All other corrections are left to be done under NLS after the file is converted.

Detailed information on the operation of TODAS is given in Appendix F.

IV GRAPHICS-ORIENTED DOCUMENT OUTPUT SYSTEM (GODOS)

GODOS has been implemented as a first-stage experimental system.

It is now possible to output an NLS file to 35-mm film via a high-resolution CRT on a CDC 3200 at another SRI computer facility. In this way drawings in the file can be converted to high-quality hardcopy, and in general a great deal of format and character-font flexibility can be gained over mechanical methods of producing hard copy.

The procedure is to output the NLS file through the output processor to a "film file" which is copied to magnetic tape. This tape is then read by the CDC 3200 and processed by a special program which controls the CRT. Thirty-five-millimeter film is automatically produced, one page per frame, and can then be used to produce high-quality Xerox copies. The total turnaround time for this process is typically on the order of hours if the volume of material is not excessive.

At present, the operating costs of GODOS are about 50 cents per frame of film and 1 or 2 cents per page of Xerox copy.

Limitations

Limitations in the accuracy of the CRT make the characters somewhat irregular, but they are highly legible.

The output display hardware uses variable-width characters. As a result, it is currently impossible to format tabular data.

The maximum total image area that can be produced on the Xerox copy is 7-1/2 by 7-1/2 inches, corresponding to about 90 characters per line and 40 lines per page. The line length is more than is needed for most purposes -- a 65-character line is standard. However, the 40-line limit is a severe limitation on the amount of information that can be included on a page.

The output processor that converts the NLS file to a film file cannot at present cope with graphic material superimposed on the text of a statement. This is regarded as a severe limitation because this intimate combination of text and graphics is a particularly attractive area for exploration as a technique for information presentation.

Detailed information on the operation of GODOS is given in Appendix G.

V NETWORK INFORMATION CENTER (NIC)

A. INTRODUCTION

The Network Information Center (NIC) is a set of services to be offered by the Augmented Human Intellect Research Center (AHIRC) of the Stanford Research Institute (SRI) to the users of the ARPA Computer Network. These services make use of the AHIRC computer, an SDS 940. User access to the NIC will be primarily through the network, but alternate means such as phone calls, letters, etc., will also be used (at least initially).

Some of the system specifications for the NIC are given in Appendix H.

A major goal of the NIC is to try to satisfy those information needs upon which the success of the "network experiment" will be most dependent. The NIC, then, is concerned with supplying information and documentation services -- as contrasted to other possible services such as project management, compiling, etc.

B. NEEDS

User needs of concern to the NIC are as follows:

(1) Creation of Documents

Users will need to create documents for use by other network users and for their own use. After a document has been created the user may wish to inspect and/or modify it.

(2) Inspection of Documents

Users will need to examine (i.e. read) documents to various depths. Such an examination may be a prelude to subsequent modification actions or other retrieval actions.

(3) Modification of Documents

Users will need to modify documents of their own creation, whether to correct errors, add, delete, change, or merge. They will also desire to modify documents created by others.

(4) Searching for Documents

The user will need to be able to scan collections of documents to search for items relevant to his work at a given moment. If a search is successful the user will desire to inspect and/or retrieve documents that he selects.

(5) Retrieval of Documents

When a user has strong interest in a document, set of documents, or sections of documents, he will need to have a copy of the material for himself. The retrieval of such information may result in copies in various forms -- e.g. hardcopy (paper, microfilm) or softcopy (computer files).

C. SERVICES

To satisfy these needs, the NIC is expected to provide the following services:

(1) Access to NIC Services

This will be initially provided via the network and via "dialup" Dataphone lines for typewriters and low-speed CRT terminals. Later, access will be provided via the network for high-performance CRT terminals.

(2) A Repository for a Collection of Documents

This collection will consist of documents contributed by network users or collected by the staff of the NIC. Elements of the collection will include research reports, user's guides, system and program descriptions, actual code, and papers of general user interest. The collection will be kept in various versions:

In hardcopy, as microfilm and paper masters held at the NIC. Replicas would be routinely distributed to each network site and other selected organizations for the users.

Special microfilm replicas of selected portions of the microfilm masters or the softcopy would be provided in microfilm form upon request.

A catalog will be one of the elements in the collection. It will list all material in the collection, whether it exists as softcopy, hardcopy, or any combination.

(3) A Documentation-Aid System

The system, oriented initially to typewriters (and later to CRTs), will incorporate text-editing and text-restructuring facilities similar to those now available in the AHIRC On-Line System (NLS).

This system is called TODAS, for Typewriter-Oriented Documentation-Aid System.

The hierarchical structure, basic to the concept of the NLS, will be available for use in all documents held by the NIC. (It may, of course,be used simply as a list of paragraphs, headings, etc.) The documentation-aid system will assist the user in generating and using documents that exploit the possibilities of this structure.

A transcription service will be available at the NIC to transcribe hardcopy documents into a softcopy version to be held in the NIC. This service will permit the initital entry into the NIC held corpus of large numbers of existing documents and newly generated documents (at least during the initial phase of the network) without unduly burdening the user. It will provide a transition interval while users become acquainted with the NLS document structure format. Ultimately, it is assumed that documents will be originally written with the use of computer aids -- either on the user's computer or with NIC's document-aid system.

(4) A Query and Search System

This system will be applicable to softcopy and hardcopy versions of the collection. By means of the "content analyzer" (a feature of the present NLS) the user will able to construct content specifications for searching the NIC collection.

By means of links between documents and within any one document (another feature of the present NLS) the user will be able to follow predetermined "trails" through the NIC collection.

Plans are being made to develop a NIC catalog, encompassing the softcopy and hardcopy (microfilm) versions of the collection.

(5) A Retrieval and Output System

This system will have online, offline, softcopy, and hardcopy (microfilm) applications.

The user will be able to request a "copy" of the document for his use. This may be a softcopy for use through NIC aids, or for use at his own computer.

The user may also make special requests for hardcopy

versions either on microfilm or on paper, both produced at the NIC. If he obtains a copy of the file at his machine he can, of course, produce his own hardcopy version.

It is expected that the user will typically desire selected "views" of the document, rather than the entire document itself. A view can be specified as to depth in the hierarchical file structure, truncation (number of lines of each statement), and section or sections of the file or files.

The view will be chosen by the user and will depend upon the depth with which he wishes to examine the document and the type of terminal at his disposal. A typewriter user will desire smaller volumes of material than will the CRT user, because of the slow speed of the terminal; the range of possible views will reflect this.

D. INITIAL PLANS

1. USERS TO BE SERVED

The initial plan covers the period from the present to December 1969. During this interval the network will be in its developmental stage and will be unavailable for general use. The parties concerned with this development will be ARPA, BBN (the network contractor), and the four initial sites (University of Utah, UCSB, UCLA, and SRI). These parties are the primary users to be served during this time with the emerging NIC services. NIC's specific service features are being oriented toward the needs of these users. Users at the other 15 sites are expected to use the NIC to a lesser degree during this period, but will become increasingly active during the latter part of 1969 and through 1970.

2. INITIAL SERVICES

The focus of the NIC will be upon the development of its basic collection of documents, a Typewriter-Oriented Documentation-Aid System (TODAS), and an early version of the on-line search and retrieval process.

Information in the initial collection is being oriented to documents pertaining to the network development, and to descriptions of systems and subystems to be available at the initial sites. This information consists of program documentation, system descriptions, user manuals, protocols and procedures, and status reports.

A NIC transcription service is already partially operational and transcribing these documents into the NIC.

A preliminary version of the Typewriter-Oriented Document Aid System (TODAS), is now in use by the transcription service, operating in the off-line mode only.

Quite early, about the 2nd quarter of 1969, the collection will be made available to the users in microfilm form. At that time a NIC system, called Graphics-Oriented Document Output System (GODOS) will enable computer-held information at the NIC to be recorded onto microfilm via a CRT.

During the third quarter of 1969 an on-line version of TODAS should be implemented to permit users with Teletype machines on the dialup telephone network to call into the NIC and execute a limited search system.

During the last quarter of 1969 the on-line version of TODAS with text-manipulation capabilities should be available.

3. PRESENT COLLECTION

The present NIC collection contains (in softcopy form) all or portions of the following documents:

(1) Half-Tone Perspective Drawings by Computer

C. Wylie, G. Romney, D. C. Evans, A. Erdahl (UTAH)

14 November 1967, Revised 12 February 1968.

(2) A FORTRAN V Interactive Graphical System

A. C. Reed, D. E. Dallin, S. T. Bennion (UTAH)

3 April 1968.

(3) GS - Graphics System

L. Copeland and C. S. Carr (UTAH)

15 November 1967.

(4) Illiac IV -- Systems Characteristics and Programming Manual

Burroughs Corporation (UI)

1 March 1968, Change 1, 12 June 1968.

(5) Procedures and Standards For Inter-Computer Communications

A. K. Bhushan and R. H. Stotz (MIT)

Reprinted from AFIPS Conference Proceedings, Volume 32, 1968.

(6) Specifications of Interface Message Processors for the ARPA Computer Network (Statement of Work Annex "B")

Advanced Research Projects Agency (ARPA)

29 July 1968.

(7) U.C.S.B. On-Line System Manual

University of California, Santa Barbara (UCSB)

1 October 1967.

(8) A Study of Computer Network Design Parameters

E. B. Shapiro (SRI)

December 1968.

(9) NIC Newsletter

NIC Staff (SRI)

16 January 1969.

(10) Network Newsletter

NIC Staff (SRI)

6 January 1969.

VI SERVICE-SYSTEM DEVELOPMENTS

A. TIME-SHARING SYSTEM

We have continued to follow closely the time-sharing system (TSS) as evolved by project GENIE at Berkeley.

We are currently running the TSS 1.96 system, which includes some improvements in user features over the earlier TSS 1.94.

This version also provides for scratch files on the disc as well as the earlier KDF disc file system.

In the last few months project GENIE has reduced its effort on the time-sharing system, and we will have to pick up any continued evolution.

B. NLS SOFTWARE

A new version of NLS was assembled early in November. This system includes most of the user features originally specified (see Section II).

Considerably more study has been done on the problem of serving NLS through the time-sharing system.

The results of these studies show that we will not be able to serve more than 6 users at a time with reasonable response under the present hardware-software system.

The study also develops a proposed solution to this situation. In particular, we plan to do some rather extensive rewriting of the NLS software and compilers, and to provide external storage for display buffers.

Extensive plans have been made for improvement and rewriting of the basic software tools for our system.

We plan to integrate the MOL, Tree Meta, and the SPLs with NLS so that source code files can be entirely in the system and compilation/assembly can be done directly from the NLS file. In addition, the compilers should run faster and compile tighter code (See Appendix I).

C. HARDWARE

Both display systems have been delivered by Tasker, but neither is accepted as yet.

Both systems are operating reasonably well and are almost up to the expected performance as discussed in the last quarterly report.

We have reached an agreement with Tasker on an acceptance test procedure, including test patterns, that will be carried out on the systems. These tests will exercise the systems at the modified performance levels and we expect that both systems will be accepted under these terms by the end of February.

The Friden keyboards have been overhauled by Friden, but they are still marginal in performance. The maintenance required to keep them in operation is much too high.

We have ordered two keyboards from Ikor for evaluation. These should be delivered in the next few weeks.

Significant improvements have been made in the video system.

We have improved the monitoring and distribution system by the addition of amplifiers, and we have added equipment for split screen and video mixing. This equipment was extremely valuable for our presentation at the FJCC, and we are now using it in a permanent setup for making movies.

We are currently making movies entirely over the video system. A film service in San Francisco will make sound films directly from a TV monitor. With the mixing and special-effects equipment, combinations of computer-generated views and live views of people and facilities are combined to produce the film.

D. COMPUTER-GENERATED SOUND

Work is progressing at a low level of effort on the implementation of a system for carrying sound signals from the computer to the individual consoles.

These signals will initially be used to carry feedback information on the state of user operations being carried out. Other uses for this information channel may be expected to arise as soon as it is available.

VII FUTURE PLANS

A. PROPOSALS FOR EXPANSION

During this last period we issued three proposals to ARPA for expansion of various aspects of our program.

We have developed specific plans for the next few months, based upon the assumption that Proposal I will be accepted. If the others are accepted, we will modify our plans accordingly.

Proposal I: Expanded NIC activity, added hardware and software to increase the capacity of our system to 12 simultaneous NLS users.

Proposal II: Interactive Display Subsystem -- for increasing the console-support capacity of our current SDS 940 system from its current 6 CRT consoles to 30, and for adding 12 more CRT consoles for local use. This would leave reserve capacity for about 5 NLS users from the network at large, or the equivalent in general Network service.

By next year we will need the capacity to handle about 24 interactive graphics terminals (for AHI and NIC staff), plus the equivalent of 5 or 6 such users as remote network service.

The modifications to our system discussed here would be an alternative to Proposal I.

The general basis for this expansion design stems from a dichotomy of service responses.

The current saturation of our 940 at about 6 to 7 users of NLS (the On-Line System) is due to excessive demands on the swapper to handle the many simple feedback responses; the processor itself is definitely not compute-bound.

Upon analysis of the response services provided by the system, there emerges a clear dichotomy that leads to some important new system-organization possibilities:

Class I service responses, requiring access to the file data or otherwise requiring the full time-sharing capability of the 940

Class II service responses, not requiring file access, but rather operating upon a relatively small amount of data (i.e., upon the "context" record for the user's

current control state and upon his display-buffer information).

This dichotomy is probably rather typical of interactive systems.

The Class II responses account for the largest number of service transactions, and their high peak rates of occurrence produce the current high, disabling ratio of swapping time to compute time.

These responses entail known, short computations, and can be serviced by simple queuing and executing to completion. In other words, they do not require the whole time-sharing machinery.

We propose to develop a special subsystem, interposed between the 940 and the display stations, that can handle work-station I/O and the Class II responses, leaving the 940 to service only the Class I responses.

It has been previously recommended that this division of servicing should obtain when we accommodate remote network users with NLS -- i.e., the host computer of the remote user should keep the control-state tables, the literal buffers, etc., and should provide the Class II responses.

Much of the Class II service is affected by the particular display devices, etc., and would need to be special to that host in any case.

At least there will have to be a conversion process in that host in order to map input devices to a standard input for us, and map a standard output description into their output-device drivers.

This plan has been put forward in a memorandum by J. F. Rulifson (SRI/AUI) and S. Carr (Utah).

Thus, this new design would offer an appealing uniformity with respect to the network, in that the 940 would service remote users via the network in exactly the same way as it would service local users via the interface subsystem.

The Class II response programs would be written in our machine-independent Special-Purpose Languages (SPLs), for which (in the Rulifson/Carr proposal) translators would be developed to compile the control programs for either our interactive-display subsystem or the remote host

computer.

Important to our approach are the interval-distribution characteristics observed with our NLS users. For Class-I responses, the mean is about 8 seconds,

With this distribution, it is estimated that the 940 could service at least 30 users with response equivalent to what it now provides when it is servicing 4 or 5 users.

In that this same type of interface subsystem could be used with less responsive time-sharing systems to allow them to accommodate some highly interactive users, the need for developing principles of design is rather general.

Also, for future such applications (both ours and others) it seems quite important that the implementation of such an "interface" subsystem be very flexible (and expandable) in terms of how many terminals it could handle.

This would mean that such a special-purpose system, fitting between a general-purpose time-sharing system and a set of interactive display terminals, could be implemented economically to serve a trial set of terminals, and could then be expanded easily, as needed (with minimum cost penalty for not beginning with the final capacity).

Proposal III: Film Output System -- for setting up a flexible, high-quality facility allowing automatic output from our 940 files onto film, in such a way that microfilm, microfiche, or print copy can be produced with a full range of integrated text and graphic content.

We are not seeking publishing-house quality in font, resolution, or stability. Such quality would be desirable in principle, but its incremental value for our purposes does not seem to justify the associated incremental costs for acquisition and operation.

We do seek a quality suitable for technical reports and documentation, but the most important added feature we seek is the capability to produce (under program control) arbitrary characters and figures at arbitrary locations on a page and to produce all the pages of a document in automatic succession.

We want to be able to handle all of the content of the documents (except photographs) within the computerized system, including the processes of composition, study, modification, and output.

We also want to use this graphical freedom of page composition to explore new techniques of hardcopy presentation for the types of material we will handle in the NIC and in our other AHI work.

We feel that it will be very valuable to the experimental scope and progress of the network if critical documentation can be flexibly and rapidly updated and then rapidly and cheaply distributed. We intend to handle text, graphs, line figures, tables and equations.

Without the proposed equipment, we would plan on using a lower quality system that is available on a closed-shop basis within SRI. We have already begun developing the programming and conventions for using this type of graphic output.

B. SOFTWARE FOR THE 940

The following software tasks are currently in the coding process.

The Tree Meta and MOL compilers are being completely rewritten. This is part of a larger program to rewrite all the system compilers, discussed in Appendix I.

Modifications to the TSS file-handling software are being made to permit permanent storage of files on the disc without the restrictions imposed by the current KDF system.

The following software tasks are currently in the design process.

The hardcopy output process and the process of viewing a file on line are both being totally reorganized and to some extent integrated with each other.

The new output and display-creation processes will involve greatly increased user control via embedded text directives (with "macrodirective" capability), simultaneous display of material from more than one file, far more sophisticated control of output formatting, and other new user features as well as improved functioning in the service-system aspects.

The current vector package in NLS (see Appendix B) will be rewritten to provide a number of new and improved features.

C. NETWORK

The Network Decode-Encode Language (DEL) is in the design stages. This is a special-purpose language for writing programs for user interaction with remote systems.

These programs, called Remote Encode Programs (REPs), have three functions: local simulation of feedback, construction of messages to go across the Network, and translation of messages received from across the Network.

For a user operating a system at a remote site, the local REP will simulate the interactive feedback of the remote system, without any communication over the Network.

The local REP will then construct hardware-independent messages from the user's control actions, and transmit them to the remote site.

The remote REP will receive these messages, translate them to the appropriate form for the system being operated, and transmit them to the system. It will then accept response messages from the system, translate them to hardware-independent form, and transmit them to the local site, where they are processed by the local REP and transmitted to the user interface.

Some details on the design of this language and its system environment are given in Appendix J.

D. NETWORK INFORMATION CENTER (NIC)

Detailed plans are now being made for implementation of the NIC; Appendix K is a planning document for this task.

Appendix A
AHI PARTICIPATION IN THE FJCC

1. INTRODUCTION

On 9 December 1968, Dr. D. C. Engelbart made a presentation to the 1968 Fall Joint Computer Conference in San Francisco. This constituted an entire Session, of which Dr. Engelbart was Chairman.

The presentation was a demonstration of the special techniques and capabilities developed by the Center; interactive computer manipulation of text, with real-time CRT display, was used as the medium for describing, demonstrating, and discussing the Center's work in developing the capabilities which were being demonstrated.

The user console, the projection equipment, and the video control equipment were located in the lecture hall; all other equipment -- the computer itself, the CRT display equipment, etc. -- remained at Stanford Research Institute in Menlo Park. Video, audio, and control information were transmitted via appropriate links leased from the telephone company.

This "computer medium" was closely coordinated with the use of speech and of advanced video techniques, and the resulting combination was communicated to the audience by means of projection television.

The television image was used to carry pictures of the computer CRT display, the faces of speakers, and equipment in the AHIRC computer room at Stanford Research Institute.

During the remaining two days of the conference, the AHI Research Center held open house in a specially prepared room at the conference.

The reaction of conference participants was highly enthusiastic, and greatly increased public and professional interest in the Center's activities is expected as a consequence.

The Center's participation in the FJCC was also a testing program for a number of new and projected developments in the Center's work. The Conference program was the culmination of several months of intensive preparatory work.

2. PREPARATION

The preparatory work fell into three major categories: new hardware and modification of existing hardware, new software, and the materials, techniques, and scenario for the actual

presentation.

a. HARDWARE PREPARATION

WORK ON VISUAL-IMAGE EQUIPMENT

The principal effort in the hardware area was concerned with the generation, processing, mixing, transmission, and projection of visual images.

Much work was done on the primary displays and display-generation equipment by both SRI and Tasker personnel, to have them in peak operating condition.

Considerable work was also done on television systems, for use in mixing various images in various ways.

Transmission of all remote information from Menlo Park to San Francisco was handled by microwave link, which entailed detailed coordination with the telephone company.

For projection of display and video images at the conference, an Eidophor television projector was obtained on loan from NASA-Ames Research Center, and considerable effort went into familiarization with this device and adjustment of its interface with the SRI equipment.

WORK ON CONSOLE EQUIPMENT

The console used for the presentation was of a new design by Herman Miller Research.

The old-style consoles used by AHIRC consist of a table, with the display monitor partially recessed in the top and the three control devices (keyboard, keyset, and mouse) at the front edge.

The new console has the monitor mounted by itself on a movable stand (wall-mounted or free-standing) and the controls mounted on a tray attached to the user's chair.

b. SOFTWARE PREPARATION

The software preparation included programming of several new features.

Provision was made to link two separate time-shared consoles

for collaborative work, by displaying the cursor spots for both consoles on each display screen. The same information is displayed on each console (by video switching, under manual control), with one user in control of the system and the other communicating with him via an audio link. The second user uses his cursor as a pointer when making reference to information on the display.

New commands were implemented to give the user direct control over the formatting of the display. This control permits detailed restructuring of the space allocations on the screen, with the primary purpose of permitting special formatting for use with video linking. Thus the user may format his screen so that all computer-generated display appears only on the left half of the screen, leaving the right half free for video of another user's face, or another user's display, etc.

The software for the first stage of a computer-generated sound system was implemented. This system will provide each console with a sound signal modulated in various ways to carry real-time information on the internal processes of the system, as relevant to the individual user.

As implemented for the FJCC, this system provided sound to only one console, and carried only limited information.

Besides these new software features, the software preparation involved accelerated development and debugging of a number of other features, including capabilities for embedding machine-executable cross-reference links in file material and an interactive keyword information-retrieval system.

c. PRESENTATION PREPARATION

In the initial stages, prparation for the session presentation consisted of extensive and detailed planning and liaison with conference officials, hardware suppliers, the telephone company, etc.

The basic plan for the presentation was to use information held in computer files as the foundation for the development of the entire presentation.

Accordingly, a complex structure of relevant files, both new and existing, was established; cross-references were inserted and a scenario was developed in terms of topics and

the files which would be used for developing each topic.

The development of the file structure and scenario made heavy use of the interactive aids for information manipulation developed in the Center's work, and this very fact was a key item in the presentation.

While this development was still taking place, a number of rehearsals were held, some of them with live audiences, both professional and lay, sophisticated and naive. Much useful feedback was obtained from these rehearsals and incorporated in changes to the scenario.

The last three rehearsals were filmed. Kinescope photography, with a special motion-picture camera aimed directly at the television screen, was found to give excellent results. These films were helpful in reviewing the performances, and also provided emergency backup for the actual presentation, where a movie projector was kept ready and a projectionist standing by in case of system failure.

3. MAIN PRESENTATION

No attempt is made here to give details on the content of the presentation. Interested persons are referred to the paper published in the Proceedings of the 1968 Fall Joint Computer Conference. A kinescope film was made of the presentation; a limited number of copies of this film will be available.

a. GENERAL DESCRIPTION

The main presentation took place in the Arena of the San Francisco Civic Auditorium. Instead of using a podium, the speaker sat at one side of the stage, with a console. A 40-foot screen was hung at the center of the stage, and the video picture was projected on this screen by the Eidophor projector. A control center was set up at the back of the Arena, along with the projector.

The speaker wore a headset and a lapel microphone; sound picked up by the microphone was amplified into the Arena, and the headset carried communications from the control crew.

The computer-generated sound was also amplified into the Arena during portions of the presentation

Two sections of the presentation were made from remote consoles in Menlo Park.

The first of these was given by J. F. Rulifson and was concerned with special software methods used in AHIRC.

Special techniques used for this section included superimposing the speaker's face upon a view of his display for general discussion purposes, and blanking out the face for detailed discussion of the displayed material.

The second remote section was by W. H. Paxton and covered information-retrieval techniques.

The beginning of this section included a demonstration of the linking capability; the projection screen carried Paxton's display with both his own tracking spot and Engelbart's, with Paxton's face shown in one corner of the screen (from which the computer display had been cleared by reformatting the display under user control).

b. AUDIENCE REACTION

An audience of perhaps 800 persons attended the main presentation. At the end of the presentation there was a sustained standing ovation.

4. OPEN HOUSE

The two-day open-house program, held in two rooms at the Civic Auditorium, consisted of informal demonstrations of AHIRC's On-Line System, informal discussions between attendees and AHIRC personnel, and display of the new Herman Miller Research equipment, set up as a complete office.

It was originally planned to hold several semiformal "minisessions" at the open house, to demonstrate and discuss individual details of the On-Line System and related systems. In practice this turned out to be impossible, because of the heavy attendance and the eagerness of attendees simply to see the total system in operation, to discuss various aspects of it, and to try it out.

Approximately 1500 persons signed a register; the estimated total attendance at the open house was 2000.

5. RESULTS

At present it is only possible to estimate some of the results of the FJCC program. The benefits fall into two categories: improved

communication and relationship with the professional computer community ("external" benefits), and benefits directly affecting the internal workings and research of the AHI Research Center ("internal" benefits).

a. EXTERNAL BENEFITS

The number of professional people who are acquainted with AHI research has been vastly increased. The professional press has shown considerable interest in AHI research as a result of the presentation. Considerable local newspaper coverage also resulted.

At a luncheon for conference participants, special official recognition was given to the session presentation.

b. INTERNAL BENEFITS

The most immediate internal benefit has been the opportunity for a general shakedown of working procedures and a considerable stimulation of new ideas for future development. Very valuable experience was also gained in several categories.

Experience in Hardware Techniques

A great deal of valuable experience was gained in the coordination and use of television equipment, not only for presentation purposes but as a technique for use in the AHI program itself. Use of such techniques as aids to interpersonal collaboration in the use of interactive display consoles is expected to become an important area of AHI research.

Experience in File Usage

The first heavy and extensive use of linked-file structures occurred in connection with the conference, and much useful information was gained.

Experience in Collaborative Techniques

Some collaborative-working techniques were tried for the first time in connection with the conference, and the experience will be of very great value since this area is one of the most important areas of planned AHI work for the future.

Appendix A
AHI PARTICIPATION IN THE FJCC

Experience in Presentation Techniques

The session presentation was by far the largest and most elaborate presentation ever mounted by AHIRC. The experience of using interactive computer techniques as a medium for communication will be of the greatest value.

Appendix B
THE NLS VECTOR PACKAGE

The vector package allows the user to create simple line drawings, with labels, as a part of his file. See the command descriptions in the NLS User's Guide or file (nlist,v:g) for how to enter the vector package in association with a particular statement.

Drawings may be output via the printer or via film; on output to other devices they disappear.

The following commands are valid within the vector package:

INSERT COMMANDS

(iv) INSERT VECTOR

Syntax: I V (CA / B / CD) CA

The bug mark is used to determine the endpoints of lines.

Each CA after the first determines a line.

Thus four CA's produce three lines, with line 1 meeting line 2 at the position of the second bug mark, and line 2 meeting line 3 at the third bug mark.

To "lift your pencil" and break the continuity of the lines type a "B" or a CD.

(il) INSERT LABEL

Syntax: I L SPACE LIT CA CA

Semantics: After typing the label, hit a CA to attach the label to the bug. The next CA fixes the label in its current position on the screen (rounded off to the nearest position that can be output on the printer).

MOVE COMMANDS

(mv) MOVE VECTOR

Syntax: M V (bug selection of vector) $(left mouse button) (bug selection of point)

Semantics: When the vector is selected, its ends are marked O and X. The end marked X will move to the point selected and the end marked O will remain fixed.

Hitting the left-hand button on the mouse wll cause the O and X to be interchanged.

The bug is then moved to the desired point and a CA hit to select the point. The "X" end of the vector will move to this point.

(ml) MOVE LABEL

Syntax: M L [1] CA CA

Semantics: When the first CA is hit, the label [1] is attached to the bug and moves with it. The next CA fixes the label in the new position.

(md) MOVE DRAWING

Syntax: M D 2$2(bug selection of point)

Semantics: The two selected points define a translation vector, and each component of the drawing is moved by this amount.

DELETE COMMANDS

(dv) DELETE VECTOR

Syntax: D V (bug selection of vector) CA

Semantics: Select the vector to be deleted and hit a CA.

(dl) DELETE LABEL

Syntax: D L [1] CA

Semantics: The label [1] is deleted.

(dd) DELETE DRAWING

Syntax: D D CA

Semantics: All vectors and all labels in the drawing are deleted. The command is used for starting over from scratch.

(t) TRANSLATE VECTOR

Syntax: T (bug selection of vector) $(left mouse button) (bug selection of point)

Semantics: This command is identical to mv in terms of actions

by the user to specify which vector and which end.

The end marked X moves to the specified new position and the end marked O moves in such a way as to preserve the length and direction of the vector.

(vv) VERTICAL

Syntax: V (bug selection of vector) $(left mouse button) CA

Semantics: When the CA is hit, the end marked X is moved horizontally so that the vector is vertical.

(h) HORIZONTAL

Syntax: H (bug selection of vector) $(left mouse button) CA

Semantics: When the CA is hit, the end marked X is moved vertically so that the vector is horizontal.

(g) GRID

Syntax: G CA

Semantics: The grid provides the user a means to draw "pretty pictures."

All positions are rounded off to the points on the grid.

The grid also places lines going through grid points such that they can be output on the printer and still look like straight lines.

The grid is either on or off; after typing "g" to get "grid" in the command feedback line, a CA causes the grid to change state.

SPACING

(sf) SPACING OFF

Syntax: S F CA

Semantics: This will set a flag that goes along with the picture telling the display creation routines not to space the statements to leave room for this picture.

(sn) SPACING ON

Syntax: S N CA

Semantics: This is the complementary command to spacing off. Since the flag is set for spacing on as the default option, this command is necessary only to change the flag back.

(a) ABORT

Syntax: A CA

Semantics: Everything that has been done in the current instance of the vector package is thrown away, the command "Vector Package" is aborted, and it is as if the command had not been given.

(f) FINISHED

Syntax: F CA

Semantics: This command returns control from the vector package to NLS proper.

Appendix C
THE INFORMATION-RETRIEVAL SYSTEM

1. INTRODUCTION

The information-retrieval system permits a user to construct a specially formatted "catalog" file, containing references to other files and capable of being reordered automatically according to some chosen set of weighted keywords. When reordered, the file lists references in order of relevance according to the choice and weighting of keywords.

Any set of statements in a file may be reordered with this system, assuming that each statement has a "name" (parenthesized first word). The specifics given in this appendix refer to the most basic way of using the system.

2. THE CATALOG FILE

The catalog file has two functioning sections: a list of file references pointing to other files, and a list of relevant keywords to be used in retrieving file references.

Other material may also be included in the file without any effect on the functioning of the retrieval system. For example, since the keyword section is to be studied directly by the user, it may be desirable to group the keyword entries into categories and separate them with headings and subheadings.

a. FILE-REFERENCE SECTION

Each file reference is a separate statement beginning with a serial number in parentheses, followed by a link pointing to the referenced file. This is followed by a list of keywords relating to the file, followed by comments on the file.

Only the first item is actually essential to the working of the system, and it need not actually be a serial number; any string of letters and/or digits enclosed in parentheses (i.e., a "statement name" as recognized by NLS) will suffice, as long as it is unique to the particular reference.

The use of serial numbers as "names" in file-reference statements, and the inclusion of the other items, are matters of convenience to the user.

b. KEYWORD SECTION

Each keyword must be a single word -- i.e., it must contain no nonprinting characters. Apart from this, it may be any arbitrary string of characters. It is convenient to use short

strings of three or four letters standing for longer words or phrases.

Each entry in the keyword section is a separate statement with the following format: first the keyword itself, in parentheses, serving as the name of the statement; then the word or phrase for which it stands, plus any comments or other information that may be desired; and finally a special code string (such as an asterisk or a dollar sign followed by a space) followed by a list of serial numbers which are the names of statements in the file-reference section. Each of these serial numbers must be enclosed in parentheses.

Examples of a short catalog file and of how it might be reordered are given at the end of this appendix.

3. KEYWORD COMMANDS

This section explains the effects of the keyword commands. Full details on the syntax and control-dialog procedures may be found in the NLS User's Guide.

The keyword commands operate upon the keywords themselves, i.e., the names of statements in the keyword section of the catalog. The commands permit the user to select keywords as relevant; assign integer weights to them; change weights; display a list of keywords that have been selected, with their weights; and produce an ordered display of the relevant file references.

a. KEYWORD SELECT COMMAND

This command is used to select a given keyword as relevant. It is automatically assigned a weight of 1.

b. KEYWORD WEIGHT COMMAND

When a keyword is selected under this command, its current weight is displayed (if it has not been previously selected, its weight is zero). The user may then type in an integer which becomes the new weight.

c. KEYWORD LIST COMMAND

This command causes display of a list of keywords with nonzero weights.

d. KEYWORD LIST WEIGHT COMMAND

This is the same as the "List" command except that the weights are shown.

e. KEYWORD FORGET COMMAND

When a keyword is selected under this command, its weight is reset to zero, just as if it had never been selected.

f. KEYWORD FORGET ALL COMMAND

This command causes all keyword weights to be reset to zero.

g. KEYWORD EXECUTE COMMAND

This command executes a program which is the heart of the system: it produces an ordered display of statements from the file-reference section of the catalog.

Each entry for a selected keyword is scanned, and the serial numbers which it contains are noted.

Each of these serial numbers is the name of a statement in the file-reference section: each of these statements is assigned a "score" equal to the weight of the keyword, and this score is accumulated with further references from other keywords.

When all of the selected keywords have been used to score the file references, the file-reference statements with nonzero scores are displayed in order of decreasing score.

4. EXAMPLE OF CATALOG FILE

a. KEYWORD SECTION

(nls) on-line system * (u1) (u2) (u3) (u4)

(ug) user guides * (u1) (u2) (u4)

(kse) keyset * (u1)

(cdp) control-dialog proc. * (u1)

(anz) content analyzer * (u2)

(fij) file jumping * (u3)

(inf) info. retrieval * (u4)

(vs) view control * (u1) (u3)

b. FILE-REFERENCE SECTION

(u1) (nlist,1:xnhj) nls,ug,vs,kse,cdp; nls user guide

(u2) (conan,1:x2bhj) anz,ug,nls; content analyzer user guide

(u3) (rlink,1:x2bhj) fij,vs,nls; link jumping and returns

(u4) (infor,1:x2bhj) inf,ug,nls; information retrieval system

5. EXAMPLE OF REORDERING

Suppose that the system is used on the catalog shown above. The user has considerable interest in file jumps, so he gives the keyword "fij" a weight of 5. He is also interested in control-dialog procedures, so he gives the keyword "cdp" a weight of 3. Finally, he is also interested in user guides, so he gives "ug" a weight of 1.

When the command Keyword Execute is given, the following scoring is done:

The keyword "fij", with a weight of 5, applies to serial number u3; therefore the statement whose name is "u3" is given a score of 5.

The keyword "cdp", with a weight of 3, applies to serial number u1; therefore the statement whose name is "u1" is given a score of 3.

The keyword "ug", with a weight of 1, applies to serial numbers u1, u2, and u4; therefore the statements whose names are "u1", "u2", and "u4" are given scores of 1 each. In the case of "u1", this is added to the previous score of 3.

The final scores are 4 for "u1", 1 for "u2", 5 for "u3", and 1 for "u4". They are then displayed as follows:

```
(u3) (rlink,1:x2bhj) fij,vs,nls; link jumping and returns
(u1) (nlist,1:xnhj) nls,ug,vs,kse,cdp; nls user guide
(u2) (conan,1:x2bhj) anz,ug,nls; content analyzer user guide
(u4) (infor,1:x2bhj) inf,ug,nls; information retrieval system
```

The user may then access the referenced files by using the Jump to Link command with the links given in the references.

Appendix D
THE NLS CONTENT ANALYZER

1. INTRODUCTION

The content analyzer feature of NLS permits the user to specify (in a special language) a pattern of content. The analyzer is compiled in real time from the user's specification, and when it is turned on (by a VIEWSPEC parameter) only statements which meet the content specification will appear on the display.

The pattern specified may be a simple one -- e.g., it may specify a string of characters that must appear somewhere in each statement to be displayed; or it may be complex -- e.g., it may specify a string, to be followed within a given number of words by another specified string, in statements which were created after a certain date by a certain author, and not containing some third specified string.

The language for specifying content patterns is simple and easy to use for simple cases, but more exacting for complex cases.

2. PATTERN-SPECIFICATION LANGUAGE

a. THE PROCESS OF SEARCHING A STATEMENT

When the content analyzer is turned on, each statement in the file is searched, character by character, for the content specified in the pattern. Normally, the search begins with the first character, but it is possible to cause the search to proceed backwards.

The analyzer uses a pointer to keep track of the search. The pointer always indicates which character is to be examined next, unless something in the pattern causes the pointer to be moved first.

At any given moment in the search process, the analyzer is searching for one of four types of content entity:

A literal string of characters, such as "abcd" or "13-x" or "ed Mat" or "memory."

A string of character-class variables; these are explained in detail further on. A string of character-class variables might specify "three digits, one after another," or "two letters, followed by any number of spaces, followed by three to five letters or digits."

The date associated with the statement. (This is not normally displayed, but every statement bears the date on which it was created or most recently modified.)

The initials associated with the statement. (This is not normally displayed, but every statement bears the initials of the user by whom it was created or most recently modified.)

All of the more complex analysis is achieved by moving the pointer according to the logic of the pattern specification.

For example, if the analyzer is to start at a given point and find either String A or String B, it first looks for string A; if String A is not found, the pointer is returned to the starting point, and a search is made for String B.

b. BASIC ELEMENTS

Every pattern ends with a semicolon.

Every pattern is made up of one or more of the basic entities listed above, combined by operators.

If the pattern (or some part of it) is to be found anywhere after the point in the statement where the search begins, it is enclosed in square brackets; otherwise it must be the first thing found.

A string of characters specified as content is enclosed in quotation marks. For convenience, if the string consists of only one character, it may be preceded by an apostrophe and the quotation marks omitted.

Examples

["memory"]; This pattern will cause display of only those statements containing the word "memory" at any point.

"inside"; This pattern will cause only statements beginning with the word "inside" to be displayed.

['3]; This pattern will cause display of only those statements containing the character "3" at any point.

Patterns like those shown in the examples above may be strung together; the significance of this is that one item is to be found after the one specified ahead of it.

Examples

["abc""def"]; This pattern specifies that the string abc immediately followed by the string def must appear somewhere in each statement to be displayed. The pattern ["abcdef"]; is exactly equivalent.

["abc"]["def"]; This pattern specifies that the string abc is to be found anywhere in the statement, and anywhere after the "c" the string def is to be found.

c. CHARACTER-CLASS VARIABLES

The character-class variables are as follows:

L means any letter

D means any digit

LD means any letter or digit

PT means any printing character (any character except space, tab, and carriage return)

SP means a space

TAB means a tab

CR means a carriage return

NP means any nonprinting character (space, tab, or carriage return)

CH means any character at all.

Examples

['.LLL'=D';]; This pattern will cause display of only those statements containing (anywhere) the following content: a period immediately followed by three letters, immediately followed by an equals sign, immediately followed by a digit, immediately followed by a semicolon.

"abcd"SPL D; This will cause display of only those statements beginning with the following content: the string abcd immediately followed by a space, immediately followed by any letter, immediately followed by any digit.

Note that a space is necessary between the L and the D because of a possible ambiguity: The pattern "abcd"SPLD; would mean "the string abcd immediately followed by a space, immediately followed by any letter or digit," because LD means any letter or digit.

d. THE DOLLAR SIGN (ARBITRARY-NUMBER CONSTRUCT)

The arbitrary-number construct, in its most general form, is m$n. The meaning is "any number from m to n of occurrences of the following entity."

When the analyzer has found n occurrences of the specified entity, it also looks ahead to see if there is another occurrence. If there is, the test is considered to have failed. In other words, the limits m and n are absolute.

Example

The pattern 5$11LD; specifies that each staement to be displayed must begin with five to eleven letters and/or digits.

A statement beginning with twelve or more letters and/or digits would be rejected by this pattern.

The m or the n, or both, may be omitted; their assumed values in this case are m=0, n=1000. For all practical purposes, then, the default value of n is "any arbitrary number," since it is very unlikely that any entity will occur 1000 times consecutively.

Examples

The pattern [7$D1$12L$5NP]; specifies that each statement to be displayed must contain the following: seven or more digits immediately followed by one to twelve letters, immediately followed by zero to five nonprinting characters.

The pattern 2$"abc"; specifies that each statement to be displayed must begin with two or more occurrences of the string abc, one after another.

e. GROUPING BY PARENTHESES

Parentheses may be used as they are in algebra to group elements. The specifications found within the parentheses are then treated as a single entity for logical purposes.

Example

[3$4(DSPL)1$2NP]; This pattern specifies that each statement to be displayed must contain the following: three or four occurrences of the string (digit space letter), immediately followed by one or two nonprinting characters.

If the parentheses were not used, the 3$4 construct would apply only to the D.

The square brackets have the same grouping effect as parentheses; however, they are not interchangeable with parentheses because they also mean that the enclosed pattern may be found anywhere after the starting point.

f. OPERATORS

The operators used for combining entities are as follows, in order of decreasing precedence (see note on precedence, below):

- (minus sign): This indicates negation. Thus -LD means a character which is not a letter or a digit.

Example: ["abc"-SP]; This pattern specifies that each statement to be displayed must contain the string abc immediately followed by some character which is not a space.

(space): This indicates concatenation. Thus "abc" "xyz"; specifies that the string abc must occur and must be immediately followed by the string xyz.

The space may be omitted unless it is necessary to prevent ambiguity. Thus "abc" "xyz"; could also be written "abc""xyz";

/ (slash): This indicates alternation. Thus SP/TAB means a character that may be either a space or a tab.

Example: 1$SP/2$3PT; This pattern specifies that each statement to be displayed must begin with either one or

more spaces, or two or three printing characters.

NOT: This indicates negation, and is the same as the minus sign except for lower precedence.

AND: This is logical intersection.

The action of the AND is to return the pointer to the beginning of the search that has just been completed.

Example: The pattern ["abc"]AND["xyz"]; causes each statement to be searched first (from the beginning) for the string abc; then, if it is found, the statement is searched again from the beginning for the string xyz. Each statement displayed will contain both strings, but the order in which they occur will be irrelevant.

Note that this is different from the pattern ["abc"]["xyz"]; because if the AND is not used, the second search is not made from the beginning but from the point just after the end of the first search. Each statement displayed will then contain both strings, but the string xyz must be somewhere after the string abc. When the AND is used, this restriction will not apply.

Note also that the pattern ["abc"AND"xyz"]; is meaningless: it specifies a string that is both "abc" and "xyz".

OR: This is the same as the slash sign except for the lower precedence.

Note on Precedence of Operators: As used here, "high precedence" means that when the pattern is parsed, the higher-precedence operators are used first in grouping the elements of the pattern. Thus a high-precedence operator has low "binding power."

Example: Consider the pattern a AND b OR c/-d AND NOT e f; where a, b, c, d, e, and f are pattern elements such as quoted strings or character-class variables.

This is grouped as follows:

The minus sign has the highest precedence, so that we have a AND b OR c/(-d) AND NOT e f;

Next is concatenation, so we have a AND b OR c/(-d) AND NOT (e f);

Next is the slash, so we have a AND b OR (c/(-d)) AND NOT (e f);

Next the NOT, giving a AND b OR (c/(-d)) AND (NOT (e f));

Finally, the AND gives (a AND b) OR ((c/(-d)) AND (NOT (e f)));.

g. DATES AND INITIALS

The dates and initials associated with each statement may be tested with the constructs .SINCE, .BEFORE, .INITIALS=, and .INITIALS#. (The symbol # is used to mean "not equal.")

The .INITIALS construct requires the following format:

.INITIALS=ABC where the string ABC is a user's initials (three initials must be given).

The .SINCE and .BEFORE constructs require the following format:

.SINCE (68/10/12 13:14) where 68 is the year, 10 is the month, 12 is the day, 13 is the hour, and 14 is the minute. The time may be eliminated by using 0:0.

Examples

.BEFORE (67/3/22 15:15) AND .SINCE (67/1/12 12:00); This pattern will cause display of only those statements bearing dates between noon of 12 January 1967 and 3:15 PM of 22 March 1967.

.SINCE (68/10/10 0:0) AND .INITIALS#DGC; This pattern will cause display of only those statements bearing dates later than 10 October 1968 and not bearing the initials DGC.

h. THE WITHIN CONSTRUCT

The WITHIN construct has the following format:

WITHIN n FIND exp1 SKIP exp2

where exp1 and exp2 are patterns and n is an integer. The search starts at the current position, and the content

specified by exp2 is skipped up to n times in a search for the content specified by exp1. If any content other than what is specified by exp2 or exp1 is found, the search fails.

Example

["write"] WITHIN 3 FIND " file" SKIP 1$NP1$PT; This pattern specifies the word "write" followed by the word "file," with up to three words intervening.

The search works as follows: after the word "write" is found, the search pointer indicates the space following the word. The exp2 pattern calls for one or more nonprinting characters followed by one or more printing characters; thus the space and the next word are skipped and the pointer again indicates a space. This skipping process is repeated up to three times, until the word "file" is found.

i. SPECIAL CONTROL OF SEARCH

The position of the search pointer can be stored and set, and the direction of search can be controlled, in order to achieve complex effects. These effects also involve the use of the IF construct (described further on), and the possibilities have been explored only superficially at present. It should be possible to create pattern expressions of great complexity which would resemble sophisticated data-processing or information-retrieval programs, but at present the techniques have not been worked out.

The position of the pointer may be stored in any one of nine buffers, P1 ... P9. This is done by writing ↑Pn, where n is some digit from 1 to 9.

The stored value in the buffer can then be decremented by writing ←Pn. The reason for doing this is that when the analyzer has found some entity, the pointer is moved to the next character position; in order to store the value of the last character actually searched, then, it is necessary to write ↑Pn←Pn.

The search pointer can then be set to the value in a buffer by writing Pn.

The search pointer can also be set to the beginning or end of a statement by writing SF(Pn) for the beginning and SE(Pn) for the end.

Note that SF and SE are functions which require a buffer value as argument; buffer values are not reinitialized after a statement has been scanned but continue to indicate the same character in the statement they were originally set to. Thus it is possible for a search to cover more than one statement.

The normal direction of scanning may be reversed by writing a less-than sign (<) and returned to the forward direction by writing a greater-than sign (>).

The left-arrow (←) used for decrementing a buffer value will increment it instead if the current scan direction is backward. Thus the effect will always be the same -- the buffer value will indicate the character just scanned.

Example

↑P1 SE(P1) < $NP -'.; This pattern causes statements to be searched backwards from the end. Only statements whose last printing character is not a period will be displayed.

The construct "↑P1" at the beginning of the pattern causes the current pointer position (which indicates the beginning of the statement) to be stored. This is simply for the purpose of having an argument for the "SE(P1)" construct, which causes the pointer to be positioned to the end of the statement. The less-than sign then causes the scan to proceed backwards; any number of nonprinting characters will be permitted, and then a character which is not a period is specified.

j. THE IF CONSTRUCT

The IF construct has the following format:

(IF relat THEN exp1 ELSE exp2)

where "relat" is a relationship between two buffer values and exp1 and exp2 are pattern expressions.

The possible relationships are as follows:

.EQ (equals)

.NE (not equal to)

.LT (less than)

.LE (less than or equal to)

.GT (greater than)

.GE (greater than or equal to).

If the specified relationship is true, exp1 is used for a test; if it is not true, exp2 is used.

Example

↑P1 SE(P1) < (['e] ↑P2←P2 AND ['t] ↑P3←P3) (IF P2 .LT P3 THEN SF(P1) > $SP "The" ELSE SF(P1) > [" if "]); This pattern imposes the following condition on statements to be displayed: If the last "e" precedes the last "t", then the first word in the statement must be "The". Otherwise, the statement must contain the word "if", enclosed by spaces. The proof is left to the reader.

k. THE .EMPTY CONSTRUCT

Whenever the analyzer makes a test, a flag is set true or false. After a statement has been tested by the complete pattern, it is displayed if the flag is true and omitted if the flag is false.

The construct .EMPTY simply sets the flag true. Conversely, the construct NOT .EMPTY (or -.EMPTY) sets the flag false.

This is useful in the IF construct, where one may simply wish to test the relationship without imposing further tests.

Example

↑P1 SE(P1) < (['e] ↑P2←P2 AND ['t] ↑P3←P3) (IF P2 .LT P3 THEN NOT .EMPTY ELSE .EMPTY); This pattern is similar to the previous example, but slightly simpler. The condition is that if the last "e" in the statement does not precede the last "t", the statement will be displayed; otherwise it will not.

The "↑P1" stores the pointer value, which inicates the beginning of the statement. The "SE(P1)" sets the pointer to the end of the statement, and the "<"

causes a backward scan. An "e" is found and its position stored in P2; then a "t" is found and its position stored in P3. The IF construct compares the values of P2 and P3: if P2 is less than P3 (i.e. if the "e" precedes the "t" in the statement), the "NOT .EMPTY" takes effect and the flag is set false, so the statement will not be displayed; if P2 is not less than P3, the ".EMPTY" takes effect, the flag is set true, and the statement is displayed.

3. PROCEDURE FOR USING CONTENT ANALYZER

A pattern may be written as text anywhere in a file. A file may thus contain any number of patterns; however, only one pattern may be compiled at a time -- i.e., when a new pattern is compiled the code created by the previous one is lost.

To compile a pattern, the command Execute Content Analyzer is used. The syntax is

```
ec [c1] CA
```

where [c1] means that a character is selected either with the mouse or by means of a pointer call, and CA means that a Command Accept key is struck.

The character selected must be either the first character of the pattern or a nonprinting character preceding the pattern, with no printing characters intervening.

Note that the last part of a pattern may thus be used as a separate pattern, if it is meaningful.

The screen will go momentarily blank with a message. If the pattern has been compiled, the message is "successful compilation"; if the pattern has an error in it which prevents it from compiling, the message is "syntax error."

Syntax errors are frequently caused by inadvertent omission of some character such as a quotation mark. Another common cause for a syntax error or a compiled pattern that does not work as expected is an error in the way that parts of the pattern are grouped. In the latter case, the problem may often be solved by insertion of parentheses.

When the pattern has been compiled, it will not go into effect until the view-control parameter "i" is placed in effect. When this has been done, the system will display only statements which

fit the pattern.

Testing of statements begins with the statement currently designated as the display start; other statements are then tested in the order in which they would appear "normally," i.e. with the analyzer off. Any other view specifications which are in effect continue to work; thus if only first- and second-level statements are being displayed, only first- and second-level statements will be tested by the analyzer.

Statements are tested until the display screen has been filled. If no statements are found that fit the pattern, the screen goes blank with the message "empty" and remains so until the analyzer is turned off or until changed view-control parameters make it possible to find a statement that fits the pattern.

Whenever the display is re-created, the testing process is repeated. Thus if a statement is edited, and the editing changes it so that it no longer fits the pattern, it will disappear from the screen.

Appendix E
THE NLS LINKING FEATURE

The command Jump to Identity is the most basic means of "moving about" within a file. A statement is selected, view-control parameters are set (if desired), and upon execution the display is re-created with the selected statement at the top and the new view parameters (if any) in effect.

Variations of this command give alternate ways of specifying the top statement. In each variation, the basic action is still the same.

This procedure has two principal limitations: the first is that only statements within the file may be selected, and the second is that the most appropriate view parameters are sometimes a matter of guesswork for the user.

To overcome these limitations, the concept of "links" has been developed.

A link is a string of text, in a special format, that specifies a point to be jumped to and a set of VIEWSPECs to be used. The specified point may be in the current file or in any other file that is accessible; the file may belong to the current user or to another user.

The basic format of a link is as follows:

(USERNAME,FILENAME,STATEMENTSPEC:VIEWSPECS)

"Username" is simply the name of a user, as recognized by the system. This is the user who "owns" the file.

If this element is omitted in the link, the owner of the file currently displayed is assumed.

"Filename" is the name of the file, as recognized by the system.

If this element is omitted in the link, it is assumed that the link refers to a place in the file currently displayed..

"Statementspec" is either a statement name or a statement number.

If this element is omitted in the link, the origin (first statement) of the file currently displayed is assumed.

"VIEWSPECs" is a string of codes for setting view-control parameters.

If this element is omitted in the link, the parameters currently in effect are assumed.

THE "JUMP TO LINK" COMMAND

The command Jump to Link is used by selecting a valid link in the file currently displayed. When the selection has been made, the username and filename (if any) are displayed as feedback at the left of the command feedback line on the display; the user executes the command by hitting Command Accept, and the display is re-created as specified by the link.

EXAMPLES

A file might contain the following text: "... see (Smith,workplan,sched:gnB) for further details."

The Jump to Link command, used on this link, would cause Smith's file "workplan" to be opened and displayed, with the statement named "sched" at the top of the display. The VIEWSPECs "gnB" would work as follows: the "g" would cause only the branch defined by the statement at the top to be displayed; the "n" would cause statement numbers to be suppressed from the display; and the "B" would cause indentation of statements according to level to be suppressed.

The user would then be working with Smith's "workplan" just as he had previously been working with his own file. He could change the VIEWSPECs, move around in the file, etc.

Now suppose that Smith's "workplan" file contains the following text: "... this is explained further in (costest,2b)." The user decides to follow this reference.

Since no username is given, the name Smith is assumed -- not the name of the actual user, but the name of the displayed file's owner. No VIEWSPECs are given, so the current values are assumed. Smith's file "costest" is loaded and displayed, with Statement 2b at the top.

This process may be continued indefinitely, as long as there are links to follow.

RETURN FROM LINK JUMPS

The system keeps track of all jumps, both within a given file (intrafile) and from one file to another (interfile). It is

possible to return to a previous view by means of special Jump commands.

RETURN FROM INTRAFILE JUMPS

The user can retrace his steps within a file by using the Jump to Return and Jump to Ahead commands.

Only five steps can be retraced in this fashion.

Whenever an intrafile jump is made, the complete specifications of the new view are added to a sequential list. This list is a history of intrafile jumps made by the user.

Exceptions: Jumps made by means of Jump to Return and Jump to Ahead are not recorded.

The Jump to Return command simply looks backward one item in the list, and re-creates that view; the Jump to Ahead command looks forward. The ends of the list are joined to each other, so that if Jump to Ahead or Jump to Return is used repeatedly, the starting point is eventually reached again.

RETURN FROM INTERFILE JUMPS

The user can retrace his steps from one file to another by using the Jump to File Return, Jump to File Ahead, Jump to File Current, and Jump to File Working Copy commands.

Again, only a limited number of steps can be retraced in this fashion. The actual number depends on the length of the links; three or four would be typical.

Whenever an interfile jump is made or a file is loaded with the Load File command, the complete specifications of the new view are added to a sequential list. This list is a history of interfile jumps seen by the user.

Exceptions: Jumps made by means of Jump to File Return, Jump to File Current, Jump to File Ahead, and Jump to File Working Copy are not recorded.

The Jump to File Return command simply looks backward one item in the list and re-creates that view; the Jump to File Ahead command looks forward. The ends of this list are not joined; thus if the user runs off either end of the list, an error is detected and the message ILLEGAL ENTITY appears.

Appendix E
NLS LINKING FEATURE

To understand the Jump to File Working Copy command, an explanation of the "WORKING COPY" file is needed.

When a file is either loaded (with the Load File command) or jumped to, the system opens it and displays it.

The system maintains a certain amount of scratch space for recording changes made by the user; these changes are not made to the file itself.

When this scratch space is exhausted, the system creates a new file called WORKING COPY, opens it, copies the altered text to it, displays it instead of the original file, and closes the original file. Thus the original file is not altered unless the command Output File is used.

When the WORKING COPY file is created, a message to that effect is displayed on the screen. If another WORKING COPY is created from another file, the old WORKING COPY is overwritten; the user, in other words, is allowed only one WORKING COPY file.

The Jump to File Working Copy command opens and displays this file. The jump is not recorded on the list.

The file WORKING COPY does not appear on the list; therefore, when the user has executed Jump to Working Copy, he cannot use Jump to Return to return to the previous view -- the result would be to return to the file displayed one step previous to the one actually desired.

In software terms, there is a pointer associated with the list, which points to the current file; the Jump to File Return and Jump to File Ahead commands simply move this pointer. However, since the WORKING COPY file is not on the list, the pointer is not moved by the Jump to File Working Copy command but remains fixed.

The return from the WORKING COPY file is made with the Jump to File Current command.

Jump to File Current cannot be used for any other purpose, since unless the WORKING COPY is being displayed it simply indicates the file currently open and displayed.

1 Appendix F
TODAS USER'S GUIDE

1a INTRODUCTION

1a1 This appendix is the User's Guide for the first-stage Typewriter-Oriented Documentation Aid System (TODAS). Certain file-manipulation operations are not explained; however, the process and procedures for creating a paper tape for input to TODAS are fully covered. It is assumed that a Teletype is used for creating the paper tape.

1b STARTING UP THE MACHINERY

1b1 Make sure you have enough paper in the Teletype and enough paper tape in the tape punch. Turn the power switch on the Teletype to the LOCAL position, and turn the punch on with the button.

1c ENTERING MATERIAL FROM THE TELETYPE

1c1 First produce a leader by hitting HERE IS a few times.

1c1a On some Teletypes, this will not work. In this case, the leader can be produced by hitting CONTROL SHIFT P 30 or 40 times.

1c2 You are now ready to start entering statements. Each statement must start with a statement number.

1c2a STATEMENT NUMBERS

1c2a1 A statement number consists of alternating sequences of numbers and letters. Each sequence is called a "field."

1c2a1a A number field is simply a number, from 1 on up. It may have any number of digits.

1c2a1b A letter field may be any single letter from a to z, or it may be made up by using the @ sign as if it were a "zero"; i.e., after z comes a@, then aa, then ab, and so forth. For most purposes, of course, single letters will be sufficient.

1c2a2 The number of fields in the statement number is called the "level" of the statement. If the statement number has only one field (which must be a number field), then it is a first-level statement. This statement is a fifth-level statement.

1c2a3 If you examine this appendix, you will see how statement numbers are used: they form a hierarchical

structure like an "outline" format. Notice that the statements are indented according to level; however, this is not always done, and it is never done when entering statements at the Teletype.

1c3 To type a statement, first type a statement number. It does not matter if mistakes are made in typing the number, as long as the level (the number of fields) is correct. Even if the level is wrong, it can be corrected at a later stage.

1c3a Only two things are absolutely essential: the statement number must begin with a digit, and it must be followed by at least one space.

1c3b If you accidentally start with a letter instead of a digit, you must abort the statement. This is explained below.

1c4 Now type the text of the statement. To end each line, use a RETURN followed by a LINE FEED. NOTE: it does not matter if, at the end of a line, you type too far and end up typing characters on top of each other. They are punched on the paper tape just the same.

1c4a To capitalize letters and to correct mistakes as you go, use the "control flags," explained below.

1c5 To end the statement, type a RETURN followed by at least two LINE FEEDs. You are then ready to start the next statement with a new statement number.

1c6 FLAG CHARACTERS

1c6a There are four characters that have special meanings. These are the backslash (-- shift L on the Teletype), the less-than sign (<), the apostrophe ('), and the dollar sign ($). These are called "flag characters."

1c6a1 The backslash is used to indicate capital letters. When it is typed immediately in front of a letter, that letter will come out as a capital in the final typing, and the backslash will disappear.

1c6a1a For example, "A QUICK BROWN FOX" will come out "A Quick Brown FOX".

1c6a2 The less-than sign is used to delete a word that has just been typed, as you are entering a statement. It "backs you up" over the last word you have typed, and also over any

spaces, LINE FEEDs, or RETURNs in front of the word. In the final typing, the word will not appear, nor will the less-than sign itself.

lc6a2a For example, "A LAZY DOG< CAT" will come out as "a lazy cat".

lc6a3 DO NOT try to use the less-than sign to correct a statement number. The system will not be able to process this correctly and the whole tape may become useless and have to be done over.

lc6a4 If you type a less-than sign after a space or a carriage-return or a tab, or any string of these characters, the system will "back up" to the end of the last word.

lc6a4a For example, "A QUICK <BROWN FOX" will come out as "A quickbrown fox".

lc6a4b Definition: A gap character (called a "GCHAR") is any space, carriage return, or tab. A GAP is any string of gap characters.

lc6a4b1 Rule:The less-than sign always causes the system to back up all the way through one GAP.

lc6a5 You can type a string of two or more less-than signs. The system will back up over that many GAPs.

lc6a5a For example, "A QUICK BROWN PIG CAT ELEPHANT<<< FOX" will come out as "A quick brown fox".

lc6a6 The apostrophe lets you use the backslash or the less-than as ordinary characters. If a backslash or a less-than sign has an apostrophe immediately in front of it, the apostrophe will disappear, the backslash or less-than sign will be typed in final typing, and the backslash or less-than sign will have no effect.

lc6a6a For example, "A '< B" will come out "a < b".

lc6a7 The apostrophe also works on itself. Two apostrophes in a row will come out as a single apostrophe, because the first one causes the second one to be taken as a normal character. The first one disappears.

lc6a7a For example, "DON''T" will come out "don't".

1c6a8 The dollar sign has special meaning only when it is the last character in a statement -- i.e., when it is immediately followed by a RETURN and at least two LINE FEEDs.

1c6a9 The effect of the dollar sign when it is the last character in a statement is to delete the statement entirely.

1c6a9a For example,

3A SOME PEOPLE ARE FICKLE

3A1 BUT GARBAGE IS $

3A1 BUT CATS ARE NOT.

will come out as follows:

3A Some people are fickle

3A1 But cats are not.

1c7 After typing the last statement, end it with a RETURN and at least two LINE FEEDs; then end the tape by punching a foot or two of leader in it, in the same way that you started the tape. Label the front end of the tape with a white pencil.

1d READING THE PAPER TAPE INTO A QED FILE

1d1 NOTE: Some of the operations in this section are not fully explained. They will be covered in a later version.

1d2 Enter at a terminal, under your own name or whatever username the file is to belong to.

1d3 Get a copy of the file /GO-TODA.

1d4 Load the front leader of the paper tape into the paper-tape reader in the control room.

1d5 Give the command GO TO FILE /GO-TODA. The system will respond with "INPUT:". Give the answer "8-LEVEL." The system will type "OUTPUT:". Give the name of the QED file you wish to store the text on. The system will type "BEGIN TRANSLATION"; then when it is done with the job it will type "END TRANSLATION".

1d6 Copy the new QED file to a disc file.

Appendix G
GODOS USER'S GUIDE

1. PURPOSE

The purpose of GODOS (Graphic Oriented Documentation Output System) is to allow NLS files with vectors embedded in the text to be output to hardcopy.

2. GENERAL DESCRIPTION

a. PASS4 PROCESS

The GODOS process is initiated using either the PASS4 Subsystem or the PASS4 output capability within NLS. (PASS4 is the general output processor that serves NLS.)

If using the PASS4 subsystem, one operates it as described in the PASS4 User's Guide. With this Subsystem QED files are used and vector information is lost in the output.

If using the PASS4 output capability within NLS, then one may obtain GODOS output in the same manner as using any other output device. For GODOS output one types "F" (for FILM) when specifying an output device.

Film files differ from those for other devices, especially with respect to variable character widths and vector and vector label information. Major differences are:

Characters produced for film are of variable width.

This creates numerous problems when attempting to have film output duplicate the page layout of the other output devices.

Tabular data are extremely difficult to produce in a readable manner, for two reasons:

First, tabs are presently set for every nth character. Since the characters vary in width, tabs will appear at different places on succeeding lines.

Second, if tab positions on the page were specified in some other manner, then one would still face the problem of lining up columnar data. Because digits are also of variable width, numbers with decimal points and commas would not necessarily appear with these characters lined up under each other on successive lines.

Vectors and vector labels are produced meaningfully only for film.

There are situations where statements with superimposed vectors will not be output properly.

Because of a line-width restriction (described below) it is advisable to produce vectors which do not appear "close to" the right-hand margin. The resulting hardcopy output from film will be magnified, and the vectors would appear too close to the right margin in the final version.

There is a maximum of only 40 lines of output per page (or film frame) for the present GODOS.

This maximum is determined by the fact that a software character set is needed for the present GODOS. This character set is defined so that the effective maximum is 40 lines if one desires readable interline spacing.

The number of characters per line is also restricted if one wishes to produce a report which approximates the present text area requirements (NASA standards for reports).

Normal hardcopy output (Xerox) using standard lens settings would produce a page with the text in an area only 7-1/2 inches in height.

It is possible to use other lens settings for the Xerox Copyflo in order to magnify this to 8-7/8 inches.

As a result, if the line width were not restricted the width of each page would also be 8-7/8 inches. Therefore, PASS4 directive default values have been adjusted so that with magnification the line width will be approximately 6-5/8 inches.

PASS4 output is to a file designated by the user. The resulting file is a normal SDS 940 file, except that "end-of-file" characters may be included because of the manner in which vector coordinates are output.

Because of the high cost of producing film it is desirable for the user to attempt to check the film file before copying it on magnetic tape. He can use two possible

methods:

First, he can use PASS4 to produce an analogous file for the printer. For this file, if he has used the standard default directives for film output, he should use the following directives for printer output: MCH=65, NTP=0, PGP=1, MLN=38, NLN=39, PLN=40.

The resulting file will only approximate the film file. An exact copy cannot be obtained at present because of the problem of variable-width characters produced for film output.

The "MCH" directive default value for film is actually 54. This value (which is converted to raster units within PASS4) is set so that a relative sized page is produced. When producing hardcopy output via the Xerox Copyflo one may then use the 35-mm lens with a setting of 15 in order to obtain an 8-1/2 by 11 inch page, with textual data appearing in an area of 6-5/8 by 8-7/8 inches. The "MCH" value of 65 for the printer is an approximation of the number of characters per line which will appear in the film file.

Second, the user can attempt to read his film output file into QED. Unfortunately, the user must become a code converter to be able to see what the file looks like, but with a few guide lines this should not be too difficult.

When the film file is read using QED, it may be impossible to obtain the entire file because vectors may have created end-of-file marks. Assuming that the file has no vectors or that the vectors have not produced these offending marks (if they have the user can at least verify his information up to the point at which the vector did terminate the file input into QED), one can proceed.

When the film file is printed using QED, the following short list of conversions should be sufficient to enable the user to "see" his file in a limited sense:

1=a, 2=b, 3=c, ..., 9=i, ↑=1, "=2, #=3, ..., @=0, P=space, K=shift-up, L=shift-down, ←=end-of-page mark, &=control character follows (in some cases).

Vector information is totally unrecognizable in

QED. The user should skip lines until he sees one which can be recognized as a possible statement.

Statement numbers and end-of-page marks can be easily identified:

Statements are normally seperated by a blank line. The heirarchical structure can be easily recognized, with lower level statements (which are indented) having leading spaces (P's) at the front of each line. Usually K's and L's will be included (for shift up and shift down -- these shifts are not designed to be analogous to standard keyboard shifts). For example: PPPK←L3P would be the beginning of statement number 1C.

Page numbers are recognized by approximately 30 spaces (P's), followed by a page number (one or more of the characters which are upper case for the digits on the typewriter keyboards), followed by a carriage return, followed by an end-of-page mark, ←. For example, [PPPPPPPPPP"(] used as a line search in QED should point to line 28 of the film document.

The PASS4 output file must then be copied to a magnetic tape (use tape 8, reel 46) using a special copy routine which will not terminate the copy process until the "actual" end-of-file mark, i.e., at least ten end-of-file marks (137b) in a row.

This routine is located in Tomlin's disc file area and is named GODGO (occasionally it is also to be found as (SYSTEM)/GO-GODOS):

When this program is available (having been retrieved from disc) then the user may operate it as he would any "go file."

At the beginning of execution the system will type "INPUT:". The user should type the name of the film file he wishes to use.

Next the system will type "OUTPUT:". The user should type a tape file name (either old or new).

If an old tape file is too short to hold the input file, a message will be typed, "OUTPUT:" will be retyped, and the user should designate a different

file.

If input and output have been properly specified, then "*****BEGIN EXECUTION*****" is typed. At the successful conclusion of the program "*****END OF PROCESS*****" is typed.

In order to place another file on tape, the user must reinitiate the above process.

The number of the resulting tape file must be noted and a data card punched for each file which is to be processed with the CDC 3200 FORTRAN program for conversion to film. The file number(s) should be punched right-justified in the first four columns of the card(s).

The data card(s) should then be placed at the end of the FORTRAN deck in numerically ascending order. Then the deck and the magnetic tape (8) should be taken to the CDC 3200.

b. FORTRAN PROGRAM TO CREATE FILM

A FORTRAN program has been developed to be run on the CDC 3200 and is the second major step in the process to produce 35-mm film and hardcopy output.

This program is designed to use as input the magnetic tape produced on the SDS 940 and data cards which are used to indicate which files on the tape are to be processed.

Output from this program is 35-mm film.

c. FILM-TO-HARDCOPY CAPABILITY

The third (and last) major step in the process is the film-to-hardcopy procedure.

The 35-mm film which is produced by the FORTRAN program via the CDC 3200 must be developed. This step can be rather rapid, depending on when the program is run and what the film developing schedule is. Essentially the film is developed using an automatic Fulton Film Processor which is now located in Building 320. This processor is operated by the service personnel connected with the CDC 3200 and under the direction of the Communication Laboratory.

The developed film is then ready to be transferred to hardcopy. There are two ways in which this can be done:

First, and probably the most useful, is to have the film automatically Xeroxed. This process produces 9-inch by 11-inch sheets of Xerox hardcopy.

Second, one can use the processed film to produce selected pages (frames) of Thermofax hardcopy.

Appendix H
NIC SYSTEM SPECIFICATIONS (PART I)

1. INTRODUCTION

This appendix proposes a set of specifications for the handling of the HCC (hardcopy collection), the FRC (film readable collection), and the MRFD (machine readable file directory). These specifications form but a part of the total set for the NIC. By far the bulk of the specifications will concern the processes associated with the transcription service, and are not considered here.

> Part II of the specifications (not given here) is concerned with TODAS (Typewriter-Oriented Documentation Aid System) and OP (Output Processor).

The HCC represents a set of documents that are physically present at the NIC in hardcopy. The FRC is a microfilm version of a portion of the MNC (master NIC collection), where the MNC consists of the HCC and the MRC (machine readable collection). This section directs attention to the FRC as it will exist in the early days of the NIC. During this time the MRC is essentially nonexistent, so the FRC contains only documents from part of the HCC.

The MRFD is the master catalog of the MNC. From the MNC one can extract information elements that can be used to construct a file directory for the FRC, the MRC, the query facility, and the retrieval facility. In part, the MRFD may contain information useful only in the administration of the NIC.

The assumed environments in which these specifications apply are two in number. The first environment is one in which the only NIC output is derived from the HCC. Here the primary concern is with the management of the documents, their selection and arrangement for filming, and the maintenance of the NIC catalog together with the construction of the index for each issue of the FRC. The second environnment is one in which the NIC output is derived from both the HCC and the MRC. The latter, through the use of the transcription service, has been derived in part from the HCC.

In the second environment the MRFD is truly the catalog for the MNC. The directory for each version of the FRC must account for this.

Though this section is addressed to the first environment, the specifications thus derived will hold for the second environment.

2. THE CATALOG -- GENERAL FEATURES

The catalog might be characterized as one of the more dynamic

elements of the NIC. It will grow and change as the MNC grows and changes. In fact, every attempt will be made to localize the effects of growth and change to the catalog, rather than extending them to the MNC.

As envisioned here, the catalog will contain entries for all documents in the HCC, whether they have been filmed or not, and all documents cited by those documents of the HCC that have been microfilmed. Included in the HCC are those entries in XDOC and the various versions of the FRC.

Entries in the catalog may vary in nature as follows:

(1) Exists by name only within the NIC; i.e., the MNC has no copy of the document

(2) Exists as a physical document

(3) Exists as a microfilmed entry in the FRC

(4) Exists in the MRC. Clearly a document can change its status with time; indeed, it can be in several states simultaneously. Thus a document may be available as hardcopy, and as microfilmed, and as transcribed to the MRC, and as transcribed from the MRC to microfilm.

It would appear less confusing to treat each form of the document as a different version, with all versions sharing a common name. This approach can give rise to a few small conflicts. For instance, the MRC version may differ in content and format from the HCC version; likewise the microfilm from the MRC can differ from the microfilm from the HCC.

A strong feature being planned for the NIC is the ability for a user to fetch a cited reference, where that reference may well be another document. To facilitate such retrieval activity, it is desirable to associate with each cited reference a unique file name. This one name should always be associated with the subject document wherever it appears, regardless of version or form of citation. Within the NIC the file name should be the key to effecting rapid retrieval of the subject document.

3. THE HARDCOPY COLLECTION

Hardcopy will be defined as encompassing printing on paper or a photograph on film. Many hardcopy documents will be in the vicinity of the NIC, but not all of the documents will be within the purview of the NIC. The documents considered within the NIC's

domain are those listed in the catalog as being retrievable upon a request of the NIC.

When a document enters the NIC's domain, several processes should be effected.

(1) A file name should be assigned to it. First, the catalog needs to be searched to determine whether the document has already been assigned a name. If it has, then the catalog entry should be appropriately updated; if not then an entry should be established.

(2) The document should be reviewed to identify citations to other documents. Once again the catalog should be searched to obtain the file names of citations already entered, and to insert the previously missing ones.

(3) An appendage should be made to the document listing the file names associated with each of the citations.

(4) The document should be filed in the physical form necessary for microfilming and reference.

(5) Keywords or other retrieval elements should be obtained from the document and also entered into the catalog.

4. THE CATALOG -- MAINTENANCE

Catalog maintenance refers to the updating of the entries through the insertion of new data, revision of existing data, and purging of old data. The possible information elements associated with each file name are as follows:

(1) File name

(2) Author's name and affiliation

(3) Title, subtitles, and other identifiers

(4) Date of publication

(5) Keywords

(6) Citations from other files

(7) Version, creation description, physical parameters (e.g., number of pages).

The completeness and size of a catalog entry will vary with the document. As a minimum an entry could consist of one element in each of items 1,2,3,4 and 7, for a "name only" document.

5. THE INDEX

The index for each version of the FRC should be derived from the catalog. The catalog is already in the system as the MRFD, so the system should be able to construct the index directly and print it out. In particular, several indices could be constructed, for example, as a permuted KWIC. It should only be necessary to specify to the system (1) the file names of the documents to be included in that version of the FRC, and (2) the order in which the documents are to appear.

The MRFD has, for each document, the number of physical pages it represents. Once the film format is established, say roll or microfiche, the system can directly enter into the index the location of each document. In the case of roll film, this location might be cassette and frame number; in the case of microfiche, this location might be card, row, and column number.

In the use of the FRC containing only HCC documents, the user is afforded only indirect links among the documents. For instance, when a document is cited, the file name having been appended, the user must enter the index by file name to obtain the document's location and then fetch that document. When a table of contents is encountered within a document, it will still refer to the physical content of the document. The user will need to search within the document on that basis, where there need not always be a simple correlation between page or section number and frame number.

Appendix I
PLAN FOR REWRITING COMPILERS

1. INTRODUCTION

The Tree Meta, MOL, and SPL compilers are in need of many changes. This appendix discusses the problems of the compilers in their current state and offers a unified solution.

We feel that a total planned rewrite of all three compilers offers the most economical long-term solution. Eventually all the things listed below must be done. If they can be accomplished with some simultaneity, all changes can be accommodated on the first pass, less time will be wasted, and the benefits of the rewrite will be available sooner.

2. CURRENT PROBLEMS AND PROPOSED SOLUTIONS

a. MOL BUGS

The most straightforward problems are the bugs in the MOL. None of these is too serious for the current MOL usage. We all just avoid using the syntactic phrases that cause the problems. This does mean, however, that the code we write does not always reflect the original, natural conception.

> This is rare enough that it certainly does not warrant a complete rewrite of the MOL; most of the bugs could be fixed by a couple of weeks' work on the current version.

A more serious problem with the MOL is the 80-character line orientation of the input routines. These programs rely on the format of QED lines; thus code that exists in NLS format must be made to look like QED format before compilation. This limits the length of NLS statements containing MOL code, and just makes everything kludgey.

b. SYMBOL PROBLEM

NLS has grown to such proportions that it nearly overflows the symbol tables of the TSS subsystems used to assemble, load, and debug it. Already it is too large to use NARP and DDT; we must use ARPAS and ODDT.

If an additive assembler were added to Tree Meta, and MOL were rewritten using the built-in assembler, this problem would completely disappear. The additive assembler would avoid the symbolic definition of the many thousand generated symbols that MOL currently produces. The only symbols defined at load time would be those specifically defined in the MOL code. This would reduce the total number from about 3000 to a few hundred.

We would design the additive assembler so that the files it produces would be NARP-DDT compatible. This would mean that we could use the new DDT and its improved debugging features.

c. SYSTEM LOAD

The load that assembling NLS currently puts on the time-sharing system (TSS) is detrimental in two ways. It kills the response of the system, eats up a lot of RAD space, and makes NLS nearly unusable while it is being done. This, in turn, makes the system programmers a little afraid to do assemblies and thus slows system design and debugging. This last problem is felt in a slow creeping way every time we put off doing an assembly for a few days because it would put such a load on the system.

The present way of assembling the system is first to compile all the files, then assemble them, and finally load them. It takes longer to assemble a file than it does to compile it; thus, getting rid of the assembly phase would cut the process in half.

Moreover, since the compilers currently spend more than half their time in the symbolic output phases, eliminating this would again cut the time in half.

Finally, the symbol-table routine wastes coniderable time in long compilations. We partially implemented a hash table in the MOL, and compilation time dropped by 1/4 for large compilations. All this means that total assembly time would drop by a factor of 5 and maybe even 10.

The major effort for the conversion to additive assemblers would be done once, in Tree Meta. The syntax for additive assembly output would closely resemble the current syntax for symbolic output.

d. COHERENT PACKAGE WITH NLS

A minor but annoying feature of the compilers as they currently stand is their kludgey interface with NLS. This is especially true when it comes to error recovery. While everything else is being rewritten, we could devise a general scheme for file processing and feedback to the user about the results of the process.

If the MOL and the SPLs were both written in Tree Meta, the code files for the system could be better organized. Each overlay could be a single file; the binary would be the result

of a single compilation.

This would simplify system assembly as well as speeding it up. Less RAD space would be needed because fewer intermediate files would be generated. Fewer symbolic and binary files would have to be saved on the disc.

Also, by having the files more closely related to program function, better use could be made of the NLS linkage commands.

e. POWERFUL SYNTAX IN MOL

A number of new features will be added to the MOL syntax. These are discussed in more detail below under MOL. The main benefit of the features is that they will make the syntax of the language closer to the intentions of the coder. This does not change anything in drastic ways; it just makes life a little better when someone is trying to figure out what a piece of code is "supposed" to do.

f. MORE DENSE SPL CODE

By rewriting the SPLs and using the features of Tree Meta, we feel that about a 20-percent reduction could be made in the number of instructions compiled. This does not affect NLS in a big way, but it would give us a little more room for expansion in some of the overlay pages that are currently over 90 percent full.

g. CONSISTENCY OF METHODOLOGY

To convert the code files to NLS and retain the current compiler systems is to do only half a job. The listings would not disappear, and the "larger NLS experiment" would not be done. To replace the listings, the code files must be coherently organized and easily accessible. For files written in MOL, this may mean experimenting with syntactic changes, and this is only practical if they are written in Tree Meta.

Eventually, we would like to work out a method of compilation that substitutes the tree structure of NLS files for the phrase structure of the MOL and SPLs. This is virtually impossible unless the MOL is in Tree Meta and the changes can be done in one central place, namely the Tree Meta library, for all the experimental compilers.

There is the vague, elusive notion of staying on top of the

design problem. The code files are becoming cumbersome to work with in their current form. Just moving them to NLS would not help much.

If, however, the syntax of the languages were more suited to NLS linkage conventions, and the files themselves were better structured, we might again reach a point of feeling that the structure is well understood, and the effect of changes in code can be properly predicted.

We have finally figured out a way of writing the parse and unparse rules for the MOL compiler in Tree Meta and not overflowing the push-down stacks during compilation. Now that we have a solution, it would be satisfying to have all of our compilers written in the same metalanguage.

3. PROPOSED CHANGES

a. TREE META

Additive Assembler

This is one of the major projects in terms of radical changes to the existing Tree Meta system. Tree Meta would be enlarged to permit either symbolic or binary output from a compilation. The binary output would be formed by making up words for a sort of backhalf processor that puts the words in the precise form necessary for DDT. Linkage for undefined labels and packing of undefined Polish expressions would be automatically handled by the backhalf.

Symbol Table

The new symbol table will use hash entry instead of the current search technique. In conjunction with the additive assembler, it will be expanded to include declaration flags, array-size parameters, and definition bits.

The new table would also reserve bits for compile-time attribute flags. This would permit a Tree Meta compiler to check declarations and give appropriate diagnostics.

Basic Recognizers

The basic recognizers will be changed to delete blanks after recognition instead of before. This will reduce the initial recognizer test, and thus the time for a failure, from the current 25 instructions to less than 5 instructions. These

failures represent about 20 percent of the runtime for a compilation.

The "TST" (literal string test) recognizer will be further improved so that a failure will average only slightly more than 3 instructions. This recognizer represents about 80 percent of the total recognizers executed. Moreover, its failure-to-success ratio is about 20 to 1.

Use of Skip Return

A new convention will be established for all the recognizers and recursive rules. The return will skip if the subroutine has been successful and not skip if it has failed. This means that the current BRANCH FALSE instruction can be eliminated. It is the shortest and yet most frequently executed POP in the Tree Meta system. It accounts for about 35 percent of the POPs executed.

Interface to NLS

Once Tree Meta has been interfaced to NLS, all the other compilers should interface automatically. It is hard to guess how long it will take to do the job, since we do not yet know what we want to do.

One suggestion is to add to NLS the ability to store a list of T-pointers, which are the result of a compilation. This list could be kept by NLS with the file until another process is performed on the file. The statements on the list would be displayed under a new VIEWSPEC parameter.

b. MOL

Rewrite in Tree Meta

The entire MOL will be written in the new Tree Meta language using the additive assembler. This project is mostly done. We have a version of the MOL written in an extended Tree Meta language using symbolic output. The code is almost complete, and we do not anticipate any new problems. Of course, the compiler cannot be checked out without a new Tree Meta because it needs features in the metalanguage which are not currently in Tree Meta.

New Features

The new MOL will have many additional features. None of

them is expensive in terms of effort or compile time. They are mostly free benefits of the use of Tree Meta for compiling.

The new compiler will allow an expression to be a statement. This will help by clearing up the meaning of many lines of current code, when an expression is forced into an ASSIGN statement even though that is not the intention of the writer.

The STORE operator, currently available only through the ASSIGN statement, will be put in as the lowest-level binding operator in an expression. This will mean that STOREs can be done during expression evaluation. This also helps conciseness and clarity.

Possible addresses will be expanded from the currently restricted set to any expression. This was always wanted, even in the original MOL specification, but was too difficult to add to the original version. The power of Tree Meta to do its top-down tree search means that the more versatile syntax can be added and tight code can still be produced for the simple cases, just as it is now.

The double branch currently compiled at the end of logical expressions will disappear. This can be done simply with the unparse rules in Tree Meta; it would have been difficult with the current MOL.

We plan to introduce a new CASE statement. It will do a single CASE based on a logical expression at the start of the CASE, rather than on a predetermined number.

Syntax will be added to simplify the use of the BRX, SKR, XMA, and register-exchange instructions. This will make all the 940 instructions available directly in the MOL (except those concerned with floating-point exponents).

Use of Additive Assembler

When the MOL is written using the additive assembler, all of the many generated labels will simply not appear in the binary file. This will mean that the number of symbols for NLS will be reduced to a manageable size. Moreover, our current kludgey way of using the "frozen" feature of ARPAS can be given up completely.

Complete Integration into MOLR

The already existing version of MOL in Tree Meta is in the MOL report file, which is in NLS format on the disc. This file represents the first attempt to integrate the actual code for a compiler into the formal and informal descriptions. This integration is only possible because the Tree Meta code for MOL is brief enough to fit in a file with the report. It may well be that this file (MOLR) could be the first realistic attempt at a single, monolithic programming and documentation structure for a large program.

Transfer of Current Code to NLS

We already have a program, PASS0, which reads an MOL program from a QED file and produces another QED file in structured-statement form. The structure is determined by a set of rules for indenting, resembling the set used by McKeeman in his UNCRUNCH program (CACM 65). We have used this program in conjunction with the NLS command Insert QED Branch with complete success, and feel that the initial transfer should be a straightforward task of only a few days.

c. SPLs

Use of Additive Assembler

When the SPL compiler and the MOL compiler are in Tree Meta, they can be rigged to output to a continuous file. This will mean that a single NLS file can contain code in both languages and still be compiled in one simple operation.

Clarity of Code in SPLs

If the SPL compiler is in Tree Meta, the parse rules will contain only parse information and node-building directions. This should make them much more readable, a feature always wanted by those who try to figure out commands of NLS by reading the code in the SPLs.

A report on the SPL is about 3/4 done (currently about 50 pages). When the SPL compiler is rewritten, the new version would be integrated into the report. This would be another large-scale attempt to do away with listing by organizing the documentation and code into an easily accessible, monolithic, structured NLS file.

4. MANPOWER ESTIMATES

To reap the full benefits from these changes, all the projects must be done as a whole. MOL and SPL cannot be rewritten without rewriting Tree Meta, and it does little good to rewrite Tree Meta only. Thus, although the estimates are broken down, the entire project must be completed to be worth the effort.

The estimate to rewrite Tree Meta and bring the report up to publishable standards is 2 man-months. The report is on the disc as a single NLS file. The new Tree Meta library and compiler will be part of the file, and the report will be synchronized with the new compiler. Most of the 2 man-months will be devoted to the new library and the additive assembler.

After the new Tree Meta is done, the MOL should only take about 1.5 more man-months. This is again for finishing the new Tree Meta version of the compiler and bringing the report file up to date and into publishable form.

Rewriting the SPLs is the simplest of the tasks. We estimate one man-month to both rewrite the compiler and finish the SPL report. About 1/3 of the time will be spent on the compiler and about 2/3 on the report.

These estimates are made in terms of time spent doing the work. Normally, the programmers within the AHI Center spend a good deal of their time debugging NLS, working on specifications and ideas for new features, and generally doing small detailed tasks not related to a specific project. With this in mind, it becomes very difficult to estimate the real time these projects will require.

Appendix J
NETWORK DECODE-ENCODE LANGUAGE

1. INTRODUCTION

This appendix is a portion of a working document for the evolution of the Decode-Encode Language (DEL). The Decode-Encode Language is a machine independent language tailored to two specific computer network tasks:

Accepting input codes from interactive consoles, giving immediate feedback, and packing the resulting information into message packets for network transmission

Accepting message packets from another computer, unpacking them, building trees of display information, and sending other information to the user at his interactive station.

2. NET STANDARD TRANSLATORS

a. INTRODUCTION

The NST library is the set of programs necessary to mesh efficiently with the code compiled at the user sites from the DEL programs it receives. The NST-DEL approach to NET interactive system communication is intended to operate over a broad spectrum.

The lowest level of NST-DEL use is direct transmission to the server-host of information in the same format that user programs would receive at the user-host.

In this mode, the NST defaults to inaction. The DEL program does not receive Universal Hardware Representation input, but input in the normal fashion for the user-host. The DEL program becomes merely a message builder and sender.

An intermediate use of NST-DEL is to have echo tables for a TTY at the user-host.

In this mode, the DEL program would run a full duplex TTY for the user.

It would echo characters, translate them to the character set of the server-host, pack the translated characters in messages, and on appropriate break characters send the messages.

When messages come from the server-host, the DEL progam would translate them to the user-host character set and print them on his TTY.

A more ambitious task for DEL is the operation of large, display-oriented systems from remote consoles over the NET.

Large interactive systems usually offer a lot of feedback to the user. The unusual nature of the feedback makes it impossible to model with an echo table, and thus a user program must be activated in a TSS each time a button state is changed.

This puts an unnecessarily large load on a TSS, and if the system is being run through the NET it could easily load two systems.

To avoid this double overloading of TSS, a DEL program will run on the user-host. It will handle all the immediate feedback, much like a complicated echo table. At appropriate button pushes, messages will be sent to the server-host and display updates received in return.

One of the more difficult, and often neglected, problems is the effective simulation of one nonstandard console on another nonstandard console.

We attempt to offer a means of solving this problem through the co-routine structure of DEL programs. For the complicated interactive systems, part of the DEL programs will be constructed by the server-host programmers. Interfaces between this program and the input stream may easily be inserted by programmers at the user-host site.

b. UNIVERSAL HARDWARE REPRESENTATION

To minimize the number of translators needed to map any facility's user codes to any other facility, there is a Universal Hardware Representation.

This is simply a way of talking, in general terms, about all the hardware devices at all the interactive display stations in the initial newtork.

For example, a display is thought of as being a square. The midpoint has coordinates (0,0), and the range is -1 to 1 on both axes. A point may now be specified to any accuracy, regardless of the particular number or density of raster points on a display.

c. INTRODUCTION TO THE NETWORK STANDARD TRANSLATOR (NST)

Suppose that a user at a remote site, say Utah, is entered in the AHI system and wants to run NLS.

The first step is to enter NLS in the normal way. At that time the Utah system will request a symbolic program from NLS.

This program is written in DEL. It is called the NLS Remote Encode Program (REP).

The program accepts input in the Universal Hardware Representation and translates it to a form usable by NLS.

It may pack characters in a buffer and may also do some local feedback.

When the program is first received at Utah, it is compiled and loaded to be run in conjunction with a standard library.

All input from the Utah console first goes to the NLS REP. It is processed, parsed, blocked, translated, etc. When the REP receives a character appropriate to its state it may finally initiate transfers to the 940. The bits transferred are in a form acceptable to the 940, and perhaps in a standard form so that the NLS need not differentiate between Utah and other NET users.

d. ADVANTAGES OF NST

After each node has implemented the library part of the NST, it need only write one program for each subsystem, namely the symbolic file it sends to each user to map the NET hardware representation into its own special bit formats.

This is the minimum programming that can be expected if each console is used to its fullest extent.

Since the NST which runs the encode translation is coded at the user site, it can take advantage of hardware at its consoles to the fullest extent. It can also add or remove hardware features without requiring new or different translation tables from the host.

Local users are also kept up to date on any changes in the system offered at the host site. As new features are added, the host programmers change the symbolic encode program. When this new program is compiled and used at the user site,

the new features are automatically included.

The advantages of having the encode translation programs transferred symbolically should be obvious.

Each site can translate any way it sees fit. Thus machine code for each site can be produced to fit that site; faster run times and greater code density will be the result.

Moreover, extra symbolic programs, coded at the user site, may be easily interfaced between the user's monitor system and the DEL program from the host machine. This should ease the problem of console extension (e.g., accommodating unusual keys and buttons) without loss of the flexibility needed for man-machine interaction.

It is expected that when there is matching hardware, the symbolic programs will take this into account and avoid any unnecessary computing. This is immediately possible through the code translation constructs of DEL. It may someday be possible to do this through program composition.

Appendix K
IMPLEMENTATION PLAN FOR THE NIC

1. INTRODUCTION

This appendix describes an implementation plan for the NIC, covering only a portion of the services that might be provided. In particular, it addresses itself to the production of a microfilm version of the MRC (Master Reference Collection) and the GODOS (Graphic-Oriented Documentation Output System).

The detailed structure of these three services is more developed than that of the other NIC services, such as TODAS (Typewriter-Oriented Documentation Aid System), query and retrieval, and NLS.

This is not to say that the gross outlines of these latter services are not known.

It is believed that the three initial services can and should continue their concurrent development.

2. MICROFILM OF THE MRC

a. THE MAJOR GOAL

A first service of the NIC should be the distribution, in microfilm form, of up-to-date versions of the hardcopy portion of the MRC.

At the present time the MRC contains user's guides of various network nodes, and network planning documents.

The microfilm material should be indexed, cross-referenced, and linked to a filing system so that material can be quickly returned.

b. TASKS TO BE PURSUED

Choice of Microfilm Format

Two formats for the microfilm should be considered -- fiche and roll.

The fiche provides about 60 frames per 4-by-6-inch card of film.

A roll provides about 3000 frames on a 100-foot 16-mm film. Long-term planning for the NIC should consider a 20-node network with each node contributing an average of 1000 pages to the MRC; thus we are considering a 20,000-page collection.

At one page per frame, in fiche form, this represents about 330 cards, while in roll-film form this is about 7 rolls.

The smaller quantum of the fiche may minimize production problems arising from routine changes in the MRC

However, there is some question regarding the ease with which a fiche collection can be used to retrieve a specific entry in the MRC.

With fiche, the present technology for low-cost systems requires a manual search for the desired card and a manual replacement into the proper location.

With roll film, the search may be eased by the use of motorized drives and automatic frame counters, though low-cost systems require the operator to be in the feedback loop.

A first task is to evaluate these two approaches.

It is recommended that a fiche viewer and a roll-film viewer (with frame counter) be obtained on a trial basis, together with some sample material.

If no clear-cut choice can be made on such a trial basis, then the hardcopy of the MRC should be experimented with.

First, the available material should be split into two parts, say 1/3 and 2/3 of the collection.

For the first 2/3, a simple index should be constructed and fiche and roll versions created and tried.

Next, to assess the effects of updating, the remaining 1/3 should be added to the microfilm version, and the index updated.

The mechanism for effecting the total update should also be studied and evaluated.

Cataloging

It is critical that a powerful catalog be established and operational when the NIC offers its initial service.

The catalog should encompass all documents within the

sphere of the MRC. As such it might contain several thousand entries, each entry being set off from all others by virtue of its identifier.

The actual form of a given document may vary with time; indeed, it may be in several forms simultaneously.

It is expected that the forms will be the following:

Machine readable

Microfilm

Hardcopy

Citation.

In its life a document might initially enter the catalog as an item cited by some other document actually in the system.

At this stage the subject document is a name only.

The document may later physically enter the system as a hardcopy item.

As interest in it rises, it could be placed on microfilm, and sometime later into the computer's files.

The structure of the catalog should be such as to accommodate a document as it assumes any and all of its forms.

Changes in a document's form should not affect the links between that document and the other documents in the MRC.

These links will point to the subject document from other documents, and should point to these other citing documents from the subject document.

These two-way links permit one to travel forward as well as backward in time.

The inclusion of entries in the catalog for cited documents that are not actually present permits all the links to be established for a document at the time it physically enters the system.

The identifier associated with a given document should be

invariant with time, even from its inception as a citation.

An identifier composed of the initial letters of the first author's surname, the year of publication, and perhaps a few auxiliary characters might suffice.

Indices and Links

The microfilmed material should contain its own index that will permit the user to use the material intelligently without any computer assistance.

There should be a master index and perhaps some subindices.

Each index could show, for each document, such familiar items as author, title, and date; it should also show the document's identifier (from the catalog) and the document's physical location on the microfilm, i.e. the roll number and frame number for roll film and the card number, row, and column for fiche.

It is highly desirable that the identifier and physical location be associated with each appearance of a citation to a document. Thus, a user of the microfilm might follow a trail of references within the filmed collection, or call for the document by identifier and use a computer terminal.

These auxiliary data might need to be appended to the hardcopy version, rather than inserted in the body of the document.

Such insertion might be difficult or undesirable to do.

The appended material might also list all documents referring to the subject document.

It is not clear that such reverse citations should be made an integral part of a document.

Such citations are likely to grow with time and result in changes to the appended portions of documents that otherwise are unchanging.

Consideration should be given to incorporation of the reverse citations in the more dynamic portions of the

file, such as the indices themselves.

Genealogies

In time there will develop films of films of hardcopy -- films of computer CRT output of system files from hardcopy, etc.

It is highly desirable that we know at all times the processes used to get to a given result.

For instance, one should be able to pick up a piece of film, find a document on it, and know all intermediate films, transformations, etc. involved in handling that document.

There is no necessity to incorporate such genealogies in the files or documents themselves.

Instruction Manual

An instruction manual should be developed for the use of the microfilm.

This manual should be a document on film and should also be available on paper.

Examples need to be constructed and a self-teaching element included.

Inventory Control

The inventory-control task is interpreted broadly to include the following:

Requesting documents for inclusion in the system

Monitoring the document inventory so that requests can be formulated

Distributing viewers and film

Collecting returnable items (cartridges, reels, etc)

Advertising the NIC's services.

3. GODOS

a. MAJOR GOALS

The major goals for GODOS are the following:

The production of film frames representing information held in system files

The production of films, observing the quantum restrictions (cards, rolls, etc.) upon user specification of the desired list of files and their ordering upon the film

The automatic generation of indices, including those documents and other films to be used in a set, where these other films may be photographs of hardcopy

The generation of title pages.

b. TASKS TO BE PURSUED

Selection and Organization of Material

Typically only a selected number of files in the MRC will be processed in a given run through GODOS.

For this to be possible it is necessary that GODOS accept, as an input, an ordered list of files.

In accordance with this list the named files should be processed in the established order.

In conjunction with that process various indices and links should be constructed.

An important element of these indices and links is information on the physical location, in the film collection, of the subject material.

Thus, an index should list the frame number (if roll film), or card-row-column number (if fiche) where each document begins.

Within a document, similar coordinate numbers should be associated with each reference to other documents or to sections within the subject document.

It is anticipated that clever processes may need to be

developed if multiple passes through the files are to be avoided.

For example, a first pass may be used to paginate the entire volume of material to be filmed, where a more than adequate number of pages are reserved beforehand for the index.

During that pass the index might be constructed.

Internal links might well require a second pass before all "targets" are specified.

It is possible that an auxiliary index could be associated with each document, especially for internal links.

This could all be based upon prepagination of each document (preestablishing a page count and relative locations of links) presupposing that a standard output format is to apply.

Format and Style Control

Various levels of format control should be effected by GODOS.

The highest level of format control would be the "volume" of text.

In the case of roll film, the "volume" could be a roll; in the case of fiche, it could be a deck of cards.

The next level of control is the document.

The concern here might be as simple as starting each document on a new frame and/or card.

It follows that each document would then terminate on a frame and/or card used only by that document.

Similar constraints might apply to chapters, sections, or other subdivisions.

Increasingly complex format rules might apply to pagination, paragraphing, and line formation.

The GODOS should provide for an open-ended set of fonts.

One or two fonts might be considered "standard," as the only ones used by a large fraction of the MRC.

For the remainder of the MRC an extensive and probably ever-growing set of fonts might be desirable.

Let it be assumed that a flexible character/vector/dot generator is used in conjunction with GODOS.

Under these conditions, essentially arbitrary fonts could be constructed.

It would then be possible to associate with each document a set of coded descriptions of any special font that it requires, and for GODOS to execute the font.

Coding embedded in the document file could indicate when GODOS should shift into and out of the special font.

Submitted by:

D. C. Engelbart, Principal Investigator

Approved by:

David R. Brown, Director
Information Science Laboratory