TM-687/008/00

Semiannual Technical Summary Report to the Director, Advanced Research Projects Agency for the period 1 January 1967 to 30 June 1967

53

TECHNICAL MEMORANDUM

(TM Series)

Semiannual Technical Summary Report

to the Director,

Advanced Research Projects Agency

for the period

1 January 1967 to 30 June 1967

SYSTEM
DEVELOPMENT
CORPORATION
2500 COLORADO AVE.
SANTA MONICA
CALIFORNIA
90406

This document was produced by SDC in performance of contract F1962867C0004 with the Electronic Systems Division, Air Force Systems Command, for the Advanced Research Projects Agency Information Processing Techniques Office.

(page 2 blank)

ABSTRACT

This report describes work done in the ARPA Information Processing Techniques Research and Laboratory Program at SDC from 1 January 1967 to 30 June 1967. Projects covered in this report include: Programming Language Development, Man-Machine Communication, Language Processing Research, and Computer Program Management.

FOREWORD

Most of the work described in this report was sponsored by the Advanced Research Projects Agency Information Processing Techniques Office, and was monitored by the Electronic Systems Division, Air Force Systems Command, under contract F1962867C0004. During the report period, some of the program was also supported by SDC's Independent Research funds.

The report is organized by major contractual tasks. A summary of the professional activities of the staff for the past six months is included at the end of the report. Documentation of the program is reported for each project. Upon request, all referenced documents will be made available to appropriate organizations.

(page 4 blank)

CONTENTS

BLANK PAGE

OVERVIEW

The past six months on the ARPA program can be characterized as a period of continued progress and transition. As in the previous reporting period, several research projects being performed under this contract have undergone significant modification. Five projects have been stopped as of the end of this period, and support for some others has now been redirected to work being done under the Advanced Development Prototype (ADP) contract. For the remaining projects, a substantial amount of effort has been devoted to reprogramming for different computers--both the Q-32 and the IBM 360.

The main areas of ARPA-supported activity under this contract were the following:

- Programming Language Development
- Man-Machine Communication
- Language Processing Research
- Computer Program Management

Within the Programming Language Development area, LISP 2 (which was developed jointly by SDC and Information International, Inc.) was terminated during this period with the completion of an initial LISP 2 compiler for the Q-32, and a complete set of design specifications for a LISP 2 system on the IBM 360 computer.

The work on meta-compilation techniques has continued to show significant promise. One of the programs produced under this project--META5--has been used successfully in a variety of applications. Further development of META5 has been transferred to the ADP contract; special emphasis is being placed on developing its data base reformatting capabilities, and on making META5 more user-oriented.

Work on META--another system being developed within the Syntax-Directed Compiler project--has been concentrated on investigating techniques for producing optimum code for powerful programming languages, as well as developing syntactic techniques for describing more powerful languages. Basically, META now consists of two components: first, a syntax pass for parsing a particular language, and second, a semantics pass for translating that language into a form usable by a particular computer.

Within the Man-Machine Communication area, work continued on time-sharing user studies, graphic input/output, and computer networks. The Time-Sharing User Characteristics study resulted in several extensive reports, one of which is a survey of the field of user performance studies. Another report produced under this study develops a model for comparing the costs of computer operating systems.

Graphic input/output project personnel are nearing completion of the transfer of their work to the Q-32 and the time-sharing environment. During the coming period, experimentation with techniques for on-line character recognition will continue, and efforts to transfer the results of this work to the IBM 360 computer will begin.

On the Computer Networks project, the TX-2 link with the Q-32 was used frequently during this period; the major use was the operation of Q-32 programs by programs running on the TX-2.

Within the Language Processing Research area, researchers continued to investigate techniques for improving the use of natural language in computer-based information systems. Under the Lexicographic Studies project, Webster's Seventh New Collegiate Dictionary was transcribed to machine-readable form, and is now being used to investigate semantic relationships among word senses by automatically processing the dictionary transcription.

Project personnel on the Transformational Grammar Tester project have developed an initial version of the tester which allows a linguist to formulate and test complex transformational grammars interactively, using a CRT display that shows structure in the form of a "tree."

A final area of work--Computer Program Management--is briefly described in this report. It was involved with studying ways to exploit recent advances in software development, particularly within military command center information systems. The study resulted in an extensive report which provides guidance on the procurement of "non-functional" software for multi-access computer systems.

PROGRAMMING LANGUAGE DEVELOPMENT

Work on the development of programming languages has been heavily weighted toward compiler construction, with a lesser amount of activity going into development of languages as such. One of the major projects, Syntax-Directed Compilers and Translators, has done a great deal of work on languages, but this work has been aimed at providing a capability to describe the compilation process, rather than to develop general-purpose programming languages. However, general-purpose languages have not been overlooked, and one project has been aimed specifically at combining some of the capabilities of a procedural language like JOVIAL with a user-oriented language like TDMS.

Work which depended on the IBM 360 computer moved, in general, somewhat slowly, but progress was good in Q-32 related projects and machine-independent studies. All the pieces of the LISP 2 system on the Q-32 are working, and a design for a 360 LISP 2 system has been prepared. The paging project produced two descriptive documents; comments on them are being sought from the programming community. The capabilities of the META and META5 systems continued to be tested with practical problems, while the ideas and principles of META have found use in other work--notably the syntax analyzer portion of the Interactive Programming Support System.

PROGRESS

1. LISP 2

LISP 2--like its predecessor, LISP 1.5--provides the programmer with a powerful means for manipulating complex data structures and performing lengthy arithmetic calculations. It extends the LISP programming language beyond that of LISP 1.5 in the following major areas: (1) The language of LISP 2 is both more convenient to use and more efficient in handling purely arithmetic operations. (2) Type declarations and new data types, including integer-indexed arrays and character strings, have been introduced to improve the efficiency of the code compiled. (3) Binary programs compiled by LISP 2 can be segmented, using "paging" techniques. (4) All LISP 2 programs and data can be dynamically relocated by automatic allocation of memory. (5) The LISP 2 system includes a time-sharing-oriented supervisor that allows incremental compilation and various other on-line debugging aids. (6) LISP 2 can be bootstrapped onto another (target) machine through the use of "core-image generation" techniques. (7) System programs are themselves written in LISP 2, and can be modified by the user. (8) The system includes five separate compilers, an assembler, and a library of subroutines, including ones for pattern-matching and for context-oriented printing. A LISP Editor (EDLISP) program is also available on the Q-32 for use with LISP 1.5 or LISP 2 programs.

A major portion of the project's time during the reporting period was spent on the design and implementation of the paging scheme for overlaying binary programs. Paging is accomplished as follows: For each routine in the computer's memory, a counter is maintained that keeps track of the time elapsed since the routine was last entered. When a request for a routine is detected, a group of functions containing that routine is brought into core. If there is not enough room for the whole group, routines in core (starting with the oldest, then the next oldest, etc.) are "thrown out."

A call to a function in secondary storage (not in core) causes entry to a trap routine, which automatically loads the called function from secondary storage. Thus it is not necessary for the calling program to know whether a function to be called is in secondary storage or in core. The secondary storage used was the disc.

The paging mechanism interacts with the "garbage collector" in helping decide upon the amount of storage to allot to various programs and data when storage is being reallocated. This reallocation is performed dynamically by moving the boundaries between various homogeneous groups of data structures. Because of the complexity of the storage management scheme and its interaction with the pager, it has been difficult to completely debug the system or "tune" the various heuristics involved with memory management.

Project personnel also completed design specifications for implementing LISP 2 on an IBM System/360 computer. The design includes multi-level input languages, a 5-pass compiler, an assembler, storage management programs, an input/output package, a core-image generator, and a swapping mechanism.

2. Syntax-Directed Compilers and Translators

The work in this area has been proceeding along two major paths. The first involves the development of META, a compiler-compiler system. This system consists of a language for describing compilers and interpreters, called "meta-language," and a program known as a "meta-compiler" which takes a compiler written in this language and produces a machine-executable version of that compiler. The second major path has led to the further development of META5, an interpretive system that consists of the META5 language, a META5 compiler, and a "pseudomachine" which is implemented on the Q-32 and IBM 360/65 computers. The META5 language allows a variety of data structures to be described and used within itself, and has operations particularly useful for data manipulation.

The primary objectives of the META work are to allow a clear and concise description of a compiler in order to reduce the costs of production, maintenance, and extensions to such programs. The meta-language employed must be able to describe the various processes which occur during compilation--to wit, syntax and semantics. The language used for describing syntax (which is similar to Backus-Naur Form) has been experimented with successfully in past years. This language is used to describe the parsing of programming languages. The remainder of the compilation process has not received as much attention.

The version of META described in the last ARPA semiannual report was powerful enough to describe itself and the machine-oriented languages for the IBM Q-32 and 360. However, it was not powerful enough to describe a full-size programming language such as JOVIAL. Also, there were inefficiencies in the GENERATOR language, which was used to describe routines that took the parsed "tree" output of the SYNTAX routines and produced machine language.

An extended META language was designed which permits the writing of the routines of a compiler in four sublanguages. The SYNTAX and GENERATOR languages are used to describe routines which recognize a source language, produce a tree form, and convert a tree form to output strings (usually machine language).

These languages are extensions of the languages described in the last semiannual report. Two new languages, called BOTTOM-TO-TOP and FUNCTIONS, were designed. These languages are used to describe routines which reconstruct the tree form of a source program from bottom to top and from top to bottom, respectively. Also, a dictionary feature has been added.

A compiler for this extended META language was written and checked out in an older version of META, which produces LISP 1.5 programs. This approach was taken for the following reasons: An intermediate form of the source program being compiled (equivalent to a tree) plays a large role in the compilation process. Parsing into tree form, output from tree form, and the optimization, analysis and synthesis of this intermediate form are of critical importance. LISP 1.5 provides efficient techniques for doing these tasks, as well as providing routines and storage allocation for managing such structures. By making use of these LISP 1.5 features, experimentation with meta-language forms can be done more easily, since the tree-handling aspects of the problem are taken care of by LISP. When the extended version of META has been evaluated and found to have met its objectives, a version will be implemented independent of LISP.

A simple compiler was written in extended META as a pilot model. The features included were beyond the scope of the earlier META's and, in fact, beyond the scope of most syntax-directed compilers. The following features were included: mixed-mode arithmetic, allocation of central registers and working storage, and optimal transfers for Boolean expressions.

This simple compiler was described at the ACM Workshop on Compiler-Building Tools held in Atlantic City on 16 April 1967.

During this period, the META5 language was extended to facilitate data manipulation and character recognition. Major improvements include:

- Notation for defining META5 character-level primitives. These permit easy definition of character sets other than those previously defined by META5.

- Number and identifier recognition primitives that do not scan trailing blanks.

- User control of column settings on input cards, including the ability to backspace on the input card image, and handle data structures in which different cards have different fixed-field formats.

- A mechanism to empty pushdown stacks or determine the depth of a stack dynamically.

META5 has been used extensively for data base reformatting. Numerous symbolic data bases produced by organizations external to SDC have been transformed into a format acceptable to existing SDC data management programs, such as LUCID, BOLD, and TRACE. The system was also used for program language legality- and syntax-checking. Using META5, programs written in a dialect of JOVIAL can be parsed to determine whether they fit the specifications of either J3 or Basic JOVIAL. If not, the system determines where the program is at variance with those specifications.

3. Compiler Techniques for Paging

Since the advent of so-called "third-generation" computers such as the IBM 360/67 and the GE 645, it has become necessary to re-evaluate program structure. A better organization of computer code is required to use the "page" and "segmentation" features of these computers efficiently. In particular, programs must be structured and arranged to minimize page-referencing. In order to avoid increasing the work required in writing code, new compiler techniques should be found which will compile ordinary higher-order language program statements into these more favorable program structures.

The purpose of this project was to determine what a compiler can accomplish through structuring and rearranging, and how such structuring can be attained. Among the areas included in the study were the optimum organization of code, methods of compiling into this form, and language forms that would be most efficient for paging.

The first phase of this project consisted of a search for techniques that were potentially useful. Some of those found were straightforward and posed no implementation problems. For example, a technique for multiple storage of a few of the most frequently used preset items was explored. At the other extreme, we investigated several newly conceived techniques whose implementation might be extremely difficult. For example, changing the order of execution of program steps would pose considerable difficulty. The techniques investigated during the first phase of this study are described in TM-3315.

The second phase of this project consisted of a more detailed study of these techniques and their implementation in a compiler. A gross design of a compiler was attempted, to show how the techniques interacted, what information needed to be generated in the various stages of compilation, and what other techniques might appear. A document (in draft form) describing this work has been prepared and will be published as TM-3315/001/00, "Compiler Symbolic Manipulations for Page-Oriented Code Optimization."

4. Data Base Oriented Programming Language

This project is concerned with combining some of the advantages of data base management systems with the power provided by procedure-oriented languages. Specifically, it is directed toward the construction of a procedure which will enable a user to prepare programs in the JOVIAL language that are compatible with a data base constructed by TDMS (Time-Shared Data Management System).

Data base management systems, such as TDMS, have several features distinctly different from procedure-oriented compiler systems, such as JOVIAL. One such feature is an English-like restricted language, which is valuable because it is user-oriented and easily learned. However, this language lacks some of the power inherent in procedure-oriented languages. Another feature--not as apparent to the user but equally significant--is the relatively flexible organization of data in storage, usually dictated by the data itself. Languages like JOVIAL, on the other hand, allow some flexibility in the process of storing data and in the hierarchical relationships between data elements, but require a relatively rigid format for the data storage, whether in core, tape, or disc.

During the reporting period, a procedure, called INQUR, was written. It is designed to afford the programmer an interface with the complex structure of a TDMS data base, and is essentially an executive procedure for calling and using TDMS procedures. The language processed by INQUR is very similar to that of the QUERY operation in TDMS. By using the TDMS procedures for data retrieval (RGET), for conditional expression evaluation (JCON), and for input/output (XIO), INQUR exposes any piece of data in the data base for free manipulation by a JOVIAL program (see Figure 1). Thus INQUR, in effect, makes the TDMS system open-ended to other programs.

INQUR is a standard JOVIAL procedure written for the IBM 360/65. Initial work was done using the available batch-processing system; the program was then modified when the time-sharing system became available. Code was written and checked out as far as possible. However, several of the TDMS procedures on which INQUR depends are not yet available. The language and usage of INQUR are described in TM-3517.

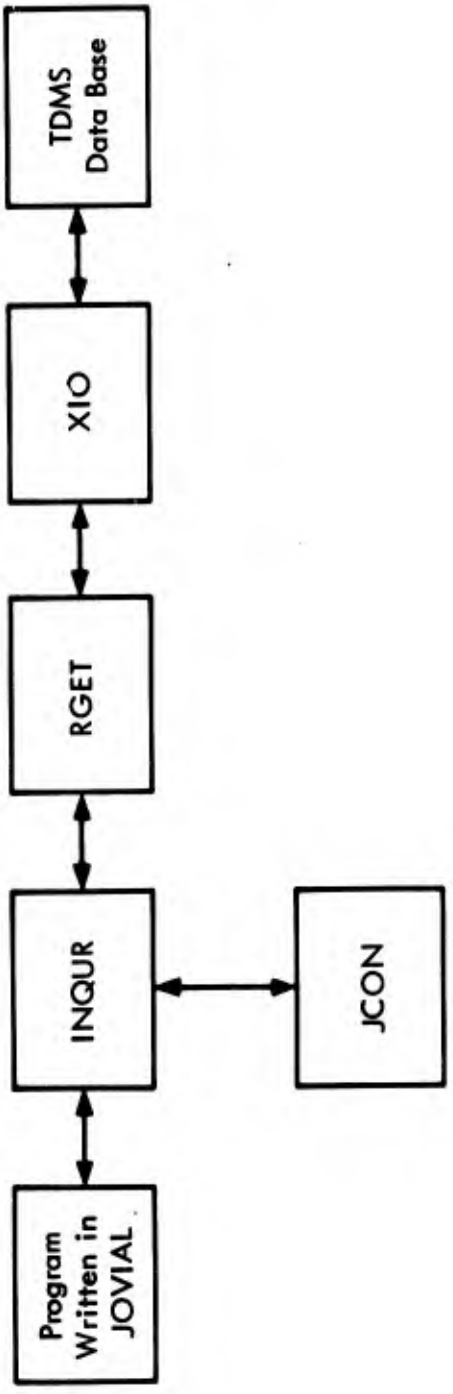

Figure 1. Interrelations Between JOVIAL Programs, INQUR, and TDMS

PLANS

1. LISP 2

There are no further plans for developing LISP 2 on the Q-32 computer. An organization external to SDC is considering the development of LISP 2 for a large, third-generation machine, and may use some of the techniques specified for LISP 2 on the IBM 360.

2. Syntax-Directed Compilers and Translators

A JOVIAL-like compiler will be written in extended META to further evaluate the language. The criteria for evaluation of this compiler will include the following: clarity and conciseness of the program; the ease with which it may be modified, that is, extended for new language features or more code optimization; and the time required to write the original compiler and make modifications to it.

Since experience with META5 has shown it to have great power for reformatting data bases, we plan to exploit this feature of META5 to aid in the conversion of symbolic data bases for the Advanced Development Prototype project. The META5 language, however, is sufficiently complicated so as to restrict its use to the dedicated systems programmer. Hence, an experimental program is being written in META5 to serve as a compiler for a special-purpose data base reformatting language that is currently under design and specification.

In this language, which will closely resemble English, the user will describe his data base. The data base compiler will then produce a program that runs under the META5 system; this program will reformat his data base into acceptable LUCID/TDMS format. The user will not be required to be familiar with LUCID/TDMS format restrictions, and in fact, he may be nearly ignorant of them. However, as an option, he may direct certain conversions, such as those involving state codes to state names, etc.

In parallel with the above activity, the present META5 language and system will be made more user-oriented by embedding a debugging capability for META5 language users in the system.

3. Compiler Techniques for Paging

With completion of the two documents describing our results, work on this project will be temporarily halted. It is expected that comments on them will be received from the programming community, and that further work can be directed by those comments.

4. Data Base Oriented Programming Language

Comments on the published description of INQUR are being collected, and the program design is being reviewed. As soon as the needed TDMS procedures become available, they will be incorporated into INQUR and the INQUR/RGET/JCON interfaces will be checked out. INQUR will also be used in checking out the RGET and JCON procedures themselves.

STAFF

LISP 2*

J. A. Barnett, Project Leader
E. Book
D. C. Crandell
D. C. Firth
R. E. Long
R. E. Martin
C. Weissman

Syntax-Directed Compilers and Translators

E. Book, Project Leader
M. Schaefer
D. V. Schorre

Compiler Techniques for Paging

R. J. Dinsmore

Data Base Oriented Programming Language

R. G. Howard

DOCUMENTATION

LISP 2

Wills, R. LISP 2 for the IBM S/360. SDC document TM-3417/000/00. 26 April 1967. 3 pp.

Establishes a document series (TM-3417) for the LISP 2 language and processor designed for the IBM S/360 computer. This series includes documents describing the syntax and semantics of the LISP 2 language, system and program design specifications, documentation standards and conventions, and user information.

*LISP 2 was developed jointly by SDC and Information International, Inc.

Wills, R. LISP 2 document conventions. SDC document TM-3417/001/00. 26 April 1967. 6 pp.

Describes conventions employed in a series of documents which specify the LISP 2 language and processor for the IBM S/360 computer. Included in this document are rules for writing syntax equations for the LISP 2 language.

Firth, D. and Abrahams, P. LISP 2 language specifications. SDC document TM-3417/200/00. 26 April 1967. 48 pp.

Describes the proposed syntax and semantics for the LISP 2 Source Language (SL) and Intermediate Language (IL) to be implemented on the IBM S/360 computer. The syntax of tokens is also included.

Barnett, J. LISP 2 compiler specifications. SDC document TM-3417/300/00. 26 April 1967. 8 pp.

Presents an overview of the LISP 2 compiler proposed for the IBM S/360 computer. It includes a brief description of the various passes of the compiler and their functions. Other documents in the series are referenced.

Barnett, J. LISP 2 compiler context resolver language and processor specifications. SDC document TM-3417/340/00. 26 April 1967. 26 pp.

Describes the language and processor required for the Context Resolver pass of the LISP 2 compiler proposed for the IBM S/360 computer. The Context Resolver (pass II of the LISP 2 compiler) is used to macro-expand Intermediate Language inputs into a list of Context-Resolved Interlude Language function definitions.

Barnett, J. LISP 2 compiler type resolver language and processor specifications. SDC document TM-3417/360/00. 26 April 1967. 16 pp.

Describes the language and processor required for the Type Resolver pass of the LISP 2 compiler proposed for the IBM S/360 computer. The Type Resolver (pass III of the LISP 2 compiler) is used to transform CRIL input into TRIL by making type conversions and confluence-point branches explicit.

Anschultz, D. LISP 2 compiler machine link specifications. SDC document TM-3417/380/00. 26 April 1967. 4 pp.

Describes the functions performed during the Machine Link pass of the LISP 2 compiler proposed for the IBM S/360 computer. The Machine Link (pass IV of the LISP 2 compiler) determines what will be accomplished by in-line code generation as opposed to function calls, and solves machine-related precision problems.

Anschultz, D. LISP 2 compiler register counter and code generator specifications. SDC document TM-3417/385/00. 26 April 1967. 12 pp.

Describes functions performed during the Register Counter and Code Generator passes of the LISP 2 compiler proposed for the IBM S/360 computer. The Register Counter (pass V of the LISP 2 compiler) counts and remembers register needs of subexpressions. The Code Generator (pass VI of the LISP 2 compiler) compiles Register-Counted Interlude Language (RCIL) into LAP assembly language.

Crandell, D. LISP 2 assembly program (LAP) specification. SDC document TM-3417/400/00. 26 April 1967. 24 pp.

Specifies the functions performed by the LISP 2 Assembly Language (AL) and processor for the IBM System 360. Included are a description of LAP address field types, LAP pseudo-instructions, and the LAP assembly processor. Syntax equations for most LAP language forms are included in an appendix.

Stygar, P. LISP 2 garbage collector specifications. SDC document TM-3417/500/00. 26 April 1967. 16 pp.

Intended to be a basis for further detailing of the work completed on storage management for LISP 2 on the IBM 360. It includes brief descriptions of housekeeping information, the four phases of garbage collection, the decentralization of primitives, the organization of the pushdown stack, and the role of various data zones. The relevance of these to the overall system is outlined, together with some of the remaining design and programming problems.

Long, R. E. LISP 2 storage management: Paging of binary programs. SDC document TM-3417/525/00. 26 April 1967. 5 pp.

Describes the paging of binary program space by the storage management functions of the LISP 2 system proposed for the IBM S/360 computer. Included are descriptions of the functions required to create, delete, load and unload library files.

Hawkinson, L. LISP 2 internal storage conventions. SDC document TM-3417/550/00. 26 April 1967. 29 pp.

Describes storage allocation conventions for the LISP 2 system proposed for the IBM S/360 computer. Core storage is to be partitioned into a large number of zones, each of which is classified according to the types of structural units that may occur within it. The configuration of each of the standard zones and its occupant units are described in detail. A scheme for the software paging of binary program space is outlined.

Stygar, P. LISP 2 storage management: The "growing pain" problem. SDC document TM-3417/575/00. 26 April 1967. 6 pp.

Presents an illustration of the "growing pain" problem, which relates to storage management functions in LISP 2. Four general approaches to a solution of the problem are presented.

Firth, D. LISP 2 input/output specifications. SDC document TM-3417/600/00. 26 April 1967. 23 pp.

Describes the capabilities required to perform input/output functions in the LISP 2 system being designed for the IBM 360/65 computer. It includes definitions of logical units and files; a discussion of file properties; a list of standard input/output functions; and specification of the monitor capabilities required for LISP 2 input/output.

Hawkinson, L. LISP 2 core image generator (CIG) specifications. SDC document TM-3417/700/00. 26 April 1967. 4 pp.

Specifies functions to be performed by the Core Image Generator (CIG) within the LISP 2 system proposed for the IBM S/360 computer. It defines the components of the CIG, how they are to be written, and alternative approaches to the core image generation process.

Syntax-Directed Compilers and Translators

Book, E. and Schorre, D. V. A simple compiler showing features of extended META. SDC document SP-2822. 11 April 1967. 7 pp.

Describes portions of a compiler written in META compiler language that shows the compilation process from source language to assembly language. The compiler produces code for a single-address machine similar to the IBM 7090. It illustrates techniques for mixed-mode arithmetic, allocation of central registers and working storage, and optimal transfers of Boolean expressions.

Schaefer, M. On the META5 language and system. SDC document SP-2823. 11 April 1967. 4 pp.

Briefly describes the META5 programming language and system. Presents the basic components of the system, an example of the language, and several application areas, including that of data base reformatting.

Schaefer, M. Recent changes to META5. SDC document N-23554. 26 January 1967. 3 pp.*

Describes changes in the language of META5 which improved the efficiency of processing and made the language easier to work with.

Compiler Techniques for Paging

Dinsmore, R. J. Compiled-program page features. SDC document TM-3315/000/00. 22 December 1966. 8 pp.

Describes "paging" optimization features that can be inserted into object programs by a compiler. The programming of such a compiler will lead to more efficient object programs for a "paging" computer.

Dinsmore, R. J. Compiler symbolic manipulations for page-oriented code optimization. SDC document TM-3315/001/00. 27 June 1967. 44 pp.

Presents a method of manipulating intermediate language program symbolism such that a final translation into computer code will produce efficient object programs for a page-oriented computer system.

Data Base Oriented Programming Language

Howard, R. G. INQUR: A procedure for retrieval from a TDMS data base. SDC document TM-3517. 15 June 1967. 27 pp.

Describes a JOVIAL procedure for the IBM 360 which retrieves data from a TDMS data base for manipulation by the calling program. The language used is similar to that of the QUERY operation of TDMS. INQUR is designed as a building block for the writing of more extensive data retrieval programs.

*This is an internal SDC document, not cleared for open publication.

MAN-MACHINE COMMUNICATION

The extremely close man-machine coupling brought about by the new computer technologies suggests the image of a right triangle, with man and machine orthogonal to one another, and communications as the hypotenuse that spans the gap. The projects reported in this section cover three aspects of this problem. The first, TSS User Characteristics, continues to explore--via experimental studies--the performance of computer users in an interactive situation. The second, Graphic Input/Output, extends the communication between man and computer via on-line character- and shape-recognition computer software and associated RAND Graphic Input Tablet/CRT display hardware. The third, Computer Networks, permits computer-to-computer communication between dissimilar, independent time-sharing systems in an attempt to span machine incompatibilities and transcontinental distances.

PROGRESS

1. TSS User Characteristics

Two projects were completed during the second half of FY 1967: one was a review and analysis of user studies in time-sharing, and the other concerned the comparative economics of computer systems. Each project is discussed in turn.

Review and Analysis of User Studies in Time-Sharing. The first study was conducted to survey and evaluate the field of user studies in time-sharing, and to develop a conceptual framework for long-range research in this area. The literature review revealed a paucity of experimental studies on user performance; a dearth of normative user statistics; the absence of any identifiable methodology or technology for investigating user behavior; and rampant speculation in the area of social effectiveness for public information utilities, rather than a planned scientific approach to the investigation of user problems in a systematic manner. The available data in the literature was most valuable in helping to define user problems and in suggesting a great variety of testable (or at least potentially testable) hypotheses on time-sharing user behavior. Although the data could be integrated to draw a suggestive portrait of the time-sharing user, the pattern was marred by great gaps in verified knowledge. On the whole, the literature was inadequate to describe a recognizable, substantive body of established findings on user behavior.

The picture is not entirely bleak, and some useful empirical leads have emerged from the small corpus of user data that is currently available. A hazy, but suggestive portrait of the time-sharing user may be discerned. The main features of this portrait are presented here:

1. More often than not, the time-sharing user will employ a terminal only once a day, and not every day; this rule is subject, however, to numerous exceptions for individual users and between different users.

2. The typical user spends between half an hour to an hour at his console at each session; this rule is also subject to great intra- and inter-subject variability.

3. Approximately 5 percent of the typical user's total working time is spent in man-computer communication at his console, whereas 95 percent of his working time is spent away from the console.

4. His median input is on the general order of one request every half-minute, whereas his average input is roughly one per minute at the terminal.

5. Half of the time he will insert a new command some 10 seconds after he has received a complete output from the computer; only on comparatively rare occasions will he wait as long as a minute before making a new request after receiving an output.

6. The ratio of human time to central processor time is on the general order of magnitude of 100:1; that is, approximately 100 seconds of elapsed human time is associated with one second of computer operating time on the user's program.

7. The computer is generally more verbose than the human; each line of human input tends to be accompanied, roughly speaking, by about two lines of computer output. New users tend to be more verbose; more terse responses occur with increasing experience.

8. Users become increasingly uncomfortable as computer response time to their requests extends beyond ten seconds, and as irregularity and uncertainty of computer response time increases.

9. Programmers seem to use, on the average, larger object programs and more computer time than non-programmers.

10. As the central system matures, and as users become more experienced, there is a tendency for object programs to grow in size and complexity, and for experienced users to require more computing time and a larger share of system capacity.

11. There are very large individual differences in human performance effectiveness against standardized on-line tasks, aften at an order of magnitude between best and poorest performers, even with relatively stratified subject samples.

12. As system load rises with increasing numbers of users, system response time increases, and, at high load-levels, both central system effectiveness and user performance tend to deteriorate.

13. User frequency distributions for most of the above parameters tend to be positively skewed; that is, most cases pile up at the low end (for example, short latencies between a completed computer output and the next human input), with a small proportion of cases tapering off at the high end (for example, very small proportions of the above latencies taking more than one minute).

A preliminary conceptual framework for user studies was constructed. In defining the area of experimental user studies, emphasis was placed on an evolutionary systems approach, based on a concept of the "user system." Although the domains of the computer and information sciences obviously mold the techniques and many of the conditions in which user studies are embedded, these disciplines, per se, were not considered as direct objects of user performance studies. Such inclusion would tend to splinter the study of users off into many directions and destroy its integrity as an area of applied research in its own right. Accordingly, experimental investigation of user performance in time-sharing was defined to include those studies concerned with techniques and findings that were experimentally observed in time-shared computing facilities. These studies should deal with the shared and verified experiences of the user community.

Proceeding from the above definition, and in the context of an evolutionary systems approach to applied scientific investigation of human performance in time-sharing, four broad conceptual areas of user system studies were delineated. These included (1) methodological, (2) normative, (3) behavioral, and (4) social effectiveness areas of inquiry. Numerous recommendations were made in relation to each of these problem areas.

1. The emphasis in methodology was on the development of a broad spectrum of experimental techniques, featuring multiple on-line monitoring of user behavior, and the parallel development of associated measures of user performance.

2. The normative area was largely oriented toward user statistics to describe performance parameters, to determine sampling characteristics (including the extent of individual user differences), and the empirical classification of users, programs, and tasks.

3. The behavioral area was concerned with hypotheses relating to user learning, interactive problem-solving, the structure and nature of individual performance differences in man-computer communication, and the impact of the interactive process on human creativity.

4. The social effectiveness area was focused on the pursuit of excellence in providing information services for the public. Under this topic, many problems potentially amenable to experimental endeavor were suggested, falling into the broad categories of user quality assurance and the pooling of social information in the public interest. The concept of the experimental computer utility, encompassing nearly all types of time-sharing user research, was singled out as the most promising vehicle for developing the techniques and gathering the knowledge necessary to serve the public interest.

The long-range program of research incorporated in the above conceptual framework for user studies is one that is oriented toward the field as a whole, not toward particular organizations or institutions.

Comparative Economics of Computer Systems. The second study in the TSS User Characteristics project was aimed at determining the comparative cost-effectiveness of four basic types of computer configurations. This study involved the development of a programmed analytic model incorporating numerous parameters bearing on the cost of representative computer systems--time-sharing, batch-processing, multiple-console batch-process, and personally operated systems. Parametric values for the model were empirically derived from a few extensively documented systems and from specially recorded data on human, machine, and software characteristics. With these data, numerous runs were conducted on the model for varying conditions, generating a variety of tradeoff curves for the four computer configurations.

The model is limited to human and equipment costs incurred in computer facility operations. It does not include human costs away from the computer or away from the time-shared console, nor does it incorporate specific types of jobs, except for a distinction between interactive versus non-interactive tasks. This investigation also encompassed experimental studies on interactive versus noninteractive debugging for programmer trainees (summarized previously), and comparative debugging performance for students under different kinds of off-line conditions. While these debugging studies clearly indicated striking performance differences under different computer configurations, the relations were too complex and the empirical data were too meager to incorporate them into the analytic cost model.

This work demonstrated the versatility of the analytic model in estimating computer system costs, for future as well as current systems. Perhaps the most valuable use of this modeling technique lies in its heuristic and educational possibilities, rather than in the specific cost figures that it produces, which are always only partial costs derived from data having high obsolescence in the swiftly changing computer field. This study makes it clear that no one computer configuration may be categorically considered as better than any other in any absolute sense; meaningful comparisons can only be made under highly specified conditions permitting quantitative measurement, and even these are subject to guarded qualification by factors not incorporated in the model as well as by differential unreliability in the empirical data.

The merit of this study lies in several departures from previous analytic investigations in this area. One is the development of a model amenable to cross-system comparisons, as opposed to the more usual case of models describing costs for only one kind of system. A second feature is the heavy and extensive use of differential empirical data from real-world computing facilities to obtain cost and effectiveness estimates as opposed to hypothetical, "armchair" exercises. A third difference is the deliberate attempt, admittedly limited, to incorporate and account for significant human parameters in the analytic model (also derived from empirical data), in contrast to the studied avoidance and gross oversimplification of human factors in traditional cost models.

2. Graphic Input/Output

The primary goal of this project is to extend the range of on-line interactive programming facilities and techniques, utilizing a RAND Graphic Input Tablet (or equivalent device), CRT displays, and their associated hardware. Central to our goal is the development of techniques for on-line character- and shape-recognition routines that function efficiently in a time-shared environment. In addition, data description, structuring, and manipulation methods must be developed that are amenable to particular applications and their environments. Such routines must be capable of being integrated into current on-line systems and extended systems.

To be generally useful, the character-recognition routines must be able to discriminate among the members of a large character set for inputs from a variety of users. Maximum payoff from such capability will be obtained in those areas where keyboards are overly restricting either in character variety, or character size and placement. For example, these areas include the input of flowchart symbols, mathematical, logical and chemical formulas (particularly when used in conjunction with automated analysis and manipulation), theorem proving, game playing, and simulation.

Use of the tablet is shown in Figures 2 and 3. In Figure 2, the user has just completed inputting the letter "B." As soon as he removes the stylus from the tablet surface, the machine-generated "B" replaces his input. Note that the output character is the same size as that input, and appears in the same position on the tablet. The hardware shown in these figures is the Grafacon Model 1010A, built by Bolt, Beranek and Newman. As a result of work done at SDC, the output characters can now be rear-projected onto the tablet surface (as shown here), producing the effect of a "live piece of paper." (In previous RAND Tablet configurations, the tablet and display surfaces were separate.)

The feasibility of recognizing characters on-line has been demonstrated. The technique consists of using each "stroke" as a basic unit. (A stroke is defined as the path of the stylus on the RAND Tablet writing surface from initial contact to lift-off.) Each stroke is processed to extract

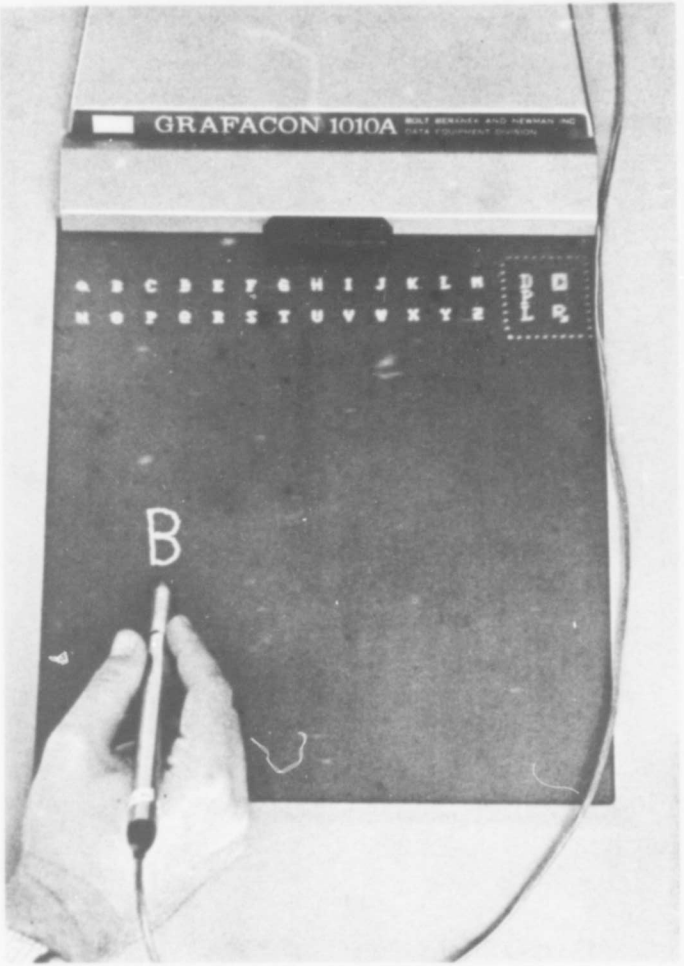

Figure 2. Handprinted Character Input on RAND Tablet

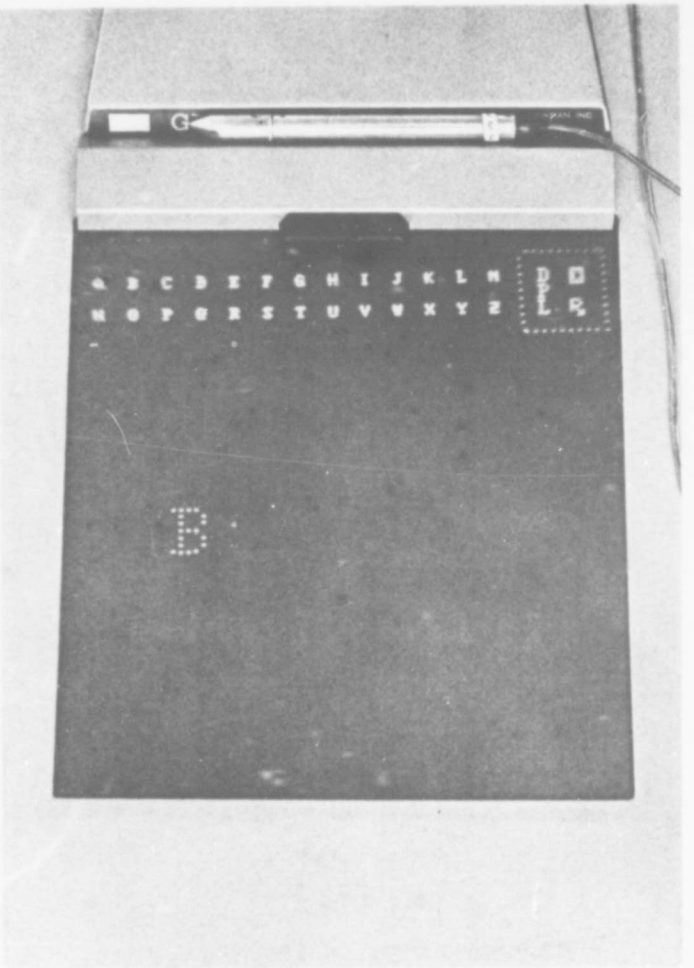

Figure 3. Computer-Generated Character Replaces Input Character

the required information, such as corners (based upon both geometry and pen velocity) and other shape-dependent features. The strokes of a multi-stroke character are properly associated by analyzing simple geometric properties that must be maintained.

Various extractors that generate shape-dependent features have been tried. They can be classified as area, local min-max and curvative extractors. These have been tested in combination with various additional analyzers that specify relationships between the shape-dependent features. It has been demonstrated that at least one stroke analyzer (feature extractor) can discriminate (with acceptable recognition rates) between 99 characters input by one individual.

Work during this reporting period was concentrated on getting a recognizer working within the constraints of the SDC Q-32 Time-Sharing System. We have also been examining new techniques and improvements to existing techniques in order to obtain more acceptable discriminators and higher recognition rates. In addition, we have been looking into the problems of dictionary optimization and purging, character-input-generated data descriptions, and data structures usable in post-recognition processes.

Several modifications had to be made to the Q-32 Time-Sharing System before we could start using a RAND Tablet in an on-line, real-time mode. These included reservation of a 1024-word buffer in Q-32 Input Memory, a system dispatcher call to write that buffer (as it is in "protected" core), and a RAND Tablet interface program in the PDP-1 input processor. (Several more iterations will be required before the modifications are finalized.)

The PDP-1 interface program provides closed-loop, real-time feedback between the RAND Tablet and the CRT display. When the RAND Tablet stylus tip switch is <u>not</u> on, the PDP-1 program displays a one-point position on the CRT, once per frame. When the tip switch is <u>on</u> (the user is writing), the PDP-1 smooths, filters, and displays the "ink" track on the CRT in real time. Smoothing (using an 8-point moving average) is optional and under control from the Q-32 user's program. The filtering program keeps account of the number of rejected points between each accepted point. This quantity is available in the display buffer and is used to compute the pen "velocity" at any given point in the stroke.

Several corner-detection schemes have been tried. The simplest uses a set of absolute criteria for determining a potential corner from "velocity valleys," and checks the geometry when a candidate is so determined. A more sophisticated version uses the average velocity of the entire stroke to determine a potential corner, and does a more thorough check of the geometry. Neither is 100 percent accurate; we are continuing to search for improvements. The PDP-1 smoothing program (which is not operational as of this writing) should improve the performance of both schemes.

In the present version of the recognizer, we have implemented only the curvature feature-extractor. Coupled with it, we have tried several methods for geometrically relating the individual features to the whole stroke. Among the methods tried were those for examining the relative position along the stroke path for the interface between features, and one for subdividing the minimum rectangle surrounding the stroke into sub-areas and noting the sub-area in which the feature interface occurred. Neither of these methods have performed as well as expected. Currently we are testing a method for computing the direction from the center of the minimum rectangle surrounding the stroke to the minimum rectangle surrounding the feature. The results to date are not conclusive, but show some promise.

A control program that is used to build the recognition dictionary is operational and provides 62 output characters, a sufficient number for testing purposes. The character set can be expanded to an arbitrary size when the need arises.

Purging the dictionary of unwanted entries can be accomplished in two ways. First, the user (dictionary builder) can specify that all dictionary items bearing a certain definition are to be deleted. This is relatively straightforward, but possibly time-consuming. Second, low-usage entries can be purged on demand if a count is kept for each use of a given definition. The recognizer has not been stable for a long enough period to try this.

Dictionary optimization involves producing more cross-linkage in the dictionary than that provided from user character samples. We have found no "sure-fire" way of doing this when there is more than one "version" of a given character in the dictionary, for example, when there are some two-stroke "A's" and some three-stroke A's in the dictionary.

Generation of data descriptions and data structures from the recognized characters poses many problems, particularly when and if the user is not required to supply additional hints, clues, or other aids. We hope to be able to solve this problem using only the general context of the application or system to which the recognizer is appended. In the instance of mathematical notation, the program will have to answer such questions as "When is a minus sign a fraction bar?" and "Is that the lower limit for the summation sign, or a free-standing equation?" Conversely, when an operator such as an integral sign or a summation sign is deleted, one should be able to remove the limits from the operator by inference, but should not necessarily remove the function to be operated upon.

3. Computer Networks

Coupling computers into a network system via commercial or military communications is as old as the SAGE system and as new as recent state and local government accounting and record-keeping systems. The new interest in computer networks is a result of the coupling of independent time-sharing

systems--either alike or dissimilar in hardware and software architecture--which permit a considerable degree of flexibility, extendability, and variability of operation over their non-time-shared counterparts. This approach to networking has been realized in the construction of a computer-controlled link between the Lincoln Laboratory TX-2 APEX System and the SDC Q-32 Time-Sharing System.

The link between the TX-2 and the Q-32 is via a Western Union 1200 BPS line. The TX-2 connects to the SDC PDP-1 computer teletype interface; in all respects (except for higher speed), the TX-2 operates like a teletype on the Q-32. Thus the TX-2 can log in, load, and execute Q-32 programs (and vice versa). This effort involved personnel from SDC, Computer Corporation of America, and Lincoln Laboratory in writing protocol documents, testing programs, and incorporating modifications to the PDP-1 and TX-2 executive programs.

Shakedown tests and demonstrations of the link have taken place during this reporting period. Multi-computer programs are running reliably on the TX-2 and Q-32 computers. Most spectacular has been the operation of an algebraic translator, T/AT, on the TX-2. The TX-2 user logs on and loads T/AT, which then proceeds to call the Q-32, log in, and load Q-32 LISP 1.5. T/AT then transmits a source program for compilation and execution by Q-32 LISP--an infix-to-prefix algebraic translator. This process takes approximately five minutes. From then on, algebraic expressions input by the TX-2 T/AT user are transmitted 3000 miles to Q-32 LISP for translation. The translation is returned to T/AT for evaluation by local software. Thus, the T/AT program is a composite of existing local and remote subroutines which run on different computer systems and use different programming languages to accomplish a given task.

A Western Union Automatic Calling Unit (ACU) was installed at SDC. A non-time-sharing utility (DIAL) was coded to check out the ACU's reliability. During April and May, tests were conducted between Lincoln Labs and SDC, with the following results:

	April		May	
	1-15	15-30	1-15	15-30
Number of calls	34	33	6	21
Percent successful	73%	85%	100%	90%
Average time to connect (sec)	19.5	19.5	19.5	21.1

Several errors were uncovered and corrected in the check-sum logic of the network's code.

Work is under way to incorporate the ACU code into the TSS PDP-1 executive, and to eliminate any remaining ambiguity of protocol between the TX-2 and the Q-32.

PLANS

1. TSS User Characteristics

With the publication of SP-2846 (see "Documentation" below), a conceptual framework now exists for long-range research in the area of the performance and characteristics of users in a time-sharing environment. These include methodological, normative, behavioral, and social effectiveness areas of inquiry. The concept of the experimental computer utility has been singled out as the most promising vehicle for further development and exploration. Within this domain, regenerative recording, and on-line survey and feedback show immediate promise.

Regenerative recording is achieved by recording all user keyboard inputs in real time. By appropriate techniques, these inputs can be replayed at a later date to regenerate the complete user-system dialog, and thus can be analyzed in accelerated time. The recordings contain a complete information store of user input characteristics. By combining pertinent system data with these user inputs, it will be possible to statistically extract such information as user error rates at character and action levels, user response time as a function of system load, and input vs. output message lengths. Other more advanced ideas permitted by synthetic recording allow experimentation with user language representations vs. keyboard error rates, or interactive dialog vs. average compute time.

The time-sharing mechanisms necessary to support regenerative recording can also be used for remote job-shop operation under time-sharing (through system command stacking) to generate synthetic recordings for subsequent regenerative playback.

On-line survey and feedback is a technique that uses the very nature of time-sharing to collect statistics about its operation and human performance. At non-critical times during system operation, e.g., at login and quit times, users are queried as to the behavior of the system: its responsiveness, ease of use, effectiveness, variety and power of its utilities. These responses are then correlated against system load to achieve a "human-engineered" balance of resources and their allocation. Such system information is given to the user as feedback to assist him in gauging future expectations of system performance.

Plans to accomplish these tasks are currently being detailed for experimentation in FY 68. With the conclusion of ARPA support in this area, further work will be continued to the extent of SDC corporate support.

2. Graphic Input/Output

During the remainder of the year we hope to work out better solutions to problems inherent in dictionary manipulation and data structuring.

The current version of the recognizer will be stabilized; we will test its capability using three project members initially.

We are constructing a testing vehicle, which--at a minimum--should allow us to manipulate a reasonable amount of text and permit us to construct an interface to one of the existing language processors on the Q-32. This will give us an opportunity to have other individuals test and evaluate the system.

When the appropriate hardware interface is completed on the IBM 360/50, we plan to replicate the (then) current version of the recognizer and its associated programs. This should broaden the potential base of users both in-house and external to SDC by providing a version that is "exportable."

3. Computer Networks

With the success of T/AT and the remote use of Q-32 LISP and TINT, sufficient interest has been generated at Lincoln Labs so that further work on the network will concentrate on more operational experience with the current capability, i.e., simple networking via high-speed teletype. A decision by Lincoln Labs has resulted in the shelving of plans to develop a display capability between the TX-2 and the Q-32. Since the bulk of APEX resources are dedicated to display programming, much of the Q-32 use of the TX-2 side of the link has been eliminated. Further plans consist chiefly of completing the ACU installation, and supporting TX-2 use of the Q-32 as needed. We plan to continue this link maintenance without ARPA support in FY 68.

STAFF

TSS User Characteristics

H. Sackman
W. J. Erikson

Graphic Input/Output

M. I. Bernstein, Project Leader
H. L. Howell
T. G. Williams

Computer Networks

N. H. Clark
C. E. Fox
W. Gardner
C. Weissman

DOCUMENTATION

TSS User Characteristics

Grant, E. E. and Sackman, H. An exploratory investigation of programmer performance under on-line and off-line conditions. IEEE Transactions on Human Factors in Electronics, Vol. HFE-8, No. 1. March 1967. pp. 33-48. Also available as SDC document SP-2581.

Describes an experiment conducted at SDC that compared the program debugging performance of programmers under conditions of on-line and off-line access to a computer. It is the first known study measuring the performance of programmers under controlled conditions for a standard task.

Sackman, H. Experimental investigation of user performance in time-shared computing systems: Retrospect, prospect, and the public interest. SDC document SP-2846. 5 May 1967. 101 pp.

Describes a survey of the field of user studies in time-sharing, and develops a conceptual framework for applied research in this area. This study traces the historical roots of user problems, and develops the need for experimental studies of user performance in time-sharing systems. A review of the literature reveals a large and growing experimental lag between the extension of information services and verified knowledge of user performance. A conceptual framework for user studies in time-sharing is constructed following three basic steps: (1) defining the field of inquiry to include experimentally derived techniques and findings on the experiences of the user community; (2) building an evolutionary systems framework for user studies, encompassing the design, development, and operation of user systems, and relating time-shared user systems to other types of computer-aided systems; and (3) classifying user problems into four broad areas--methodological, normative, behavioral, and social effectiveness. Numerous problems, hypotheses, and recommendations for experimental investigation of user performance are made for each of these four categories. The study concludes with a plea for interdisciplinary applied research to meet the imminent challenge of the public information utility.

Erikson, W. J. An analytical cost comparison of computer operating systems. SDC document TM-3525. 30 June 1967. 214 pp.

Describes research undertaken for the purpose of answering questions concerning the advantages and disadvantages of time-sharing. To accomplish this, the general problem of evaluating computer system performance is first addressed. In this evaluation, no attempt is made to compare computers supplied by different manufacturers. Rather, general system characteristics are specified that include the computer and its operating system, and users and their jobs. The main emphasis is placed upon the operating system; the effects of having different computers, users, or jobs are treated as parameters. The most important evaluation criterion is considered to be cost, which includes both user cost and computer system cost.

Quantitative models are developed that describe computer center users, the programs they run, and the different operating systems they might use. These include time-sharing, batch-processing, multiple-console batch-processing, and personally operated systems. Empirical data about these same subjects are used as inputs to the models in an attempt to determine how well the various computer systems might serve different sets of users.

The estimates of the model parameters were obtained from a few extensively documented systems, experiments, analyses of historical data, and system-recorded data. With these data, a few systems are analyzed in depth both to demonstrate the application of the analysis technique and to arrive at some conclusions regarding the relative economics of the systems from which data are derived.

The research concentrates on the performance of a single computer system that is configured to run with one of the four operating systems under study. Total costs are computed for a group of users with a given job mix who use each of the systems to process their jobs.

Answers are not given to all of the questions that might arise--either for the manager or for researchers--during computer system performance evaluation. Based upon the simple models used, some limited conclusions may be drawn about which system is best. The most important result of this study may be the approach that was used and the promise that computer-assisted models appear to hold for the development of meaningful computer system performance evaluations in the future.

LANGUAGE PROCESSING RESEARCH

The ARPA-sponsored language processing research program at SDC included three projects: lexicographic studies, the development of a transformations-testing system, and the study of English query subset languages. The common theme of the three studies is to develop information and tools to aid computational linguists and language processors in their attempts to use English or subsets of English as command or query languages in computing systems.

The lexicographic study has been concerned with converting Webster's Seventh New Collegiate Dictionary to computer-readable form and developing processing methods and a processor for conducting semantic studies of its contents. The aime of the transformations-tester project is to develop an interactive computer system that can aid a linguist in formulating and testing sets of complex transformational rules for syntactic analysis of English. The query subsets study attempted to discover parameters that would predict the effectiveness and error-proneness of any subset of English to be used as a query language in a data management system.

The lexicographic study has achieved its first goal of making the dictionary computer-readable. A preliminary design for a general-purpose processor has been developed, and several papers have been written outlining patterns of relationships among defining senses for words. The transformations-tester study has developed an effective, interactive system that allows a linguist to formulate and test complex transformational grammars. The query subsets study achieved only its first goals of developing a grammar and measures of preference for the forms of English questions, before it was discontinued.

Detailed statements of the progress and plans for each of these projects are included in the following sections.

PROGRESS

1. Lexicographic Studies

The overall aim of this project is to enlist the aid of computers in describing the vocabulary structures of natural languages. It has long been recognized that word senses reciprocally delimit each other within a language. Thus, in carrying out a semantic analysis of a given word (or, more properly, a morpheme) in some language, it is important to take account of all the other words in that language that are significantly related semantic lly to the given word. However, to prepare an exhaustive list of these other words by hand is an almost hopeless task, at least for any of the major languages, because of the sheer size of their vocabularies. In principle, no such list could be considered exhaustive unless every word in the language had been considered for inclusion or rejection with respect to the given word. In addition, there are problems in deciding exactly what should count as a significant semantic relation between a pair of words.

We have been immediately concerned with the vocabulary structure of English as manifested in Webster's Seventh New Collegiate Dictionary, henceforth referred to as W7. We have been laying the groundwork for a program system that can be used in either an automatic or an interactive, semiautomatic mode to specify the semantic field(s) of any randomly chosen word by processing a Hollerith transcription of W7. For each sense of the word, a semantic field will be specified by locating the sense within a network of other word-senses having specified relationships to the given sense (and to each other). To achieve this goal, two major research tasks have been undertaken:

1. Developing criteria for judging whether a given pair of word-senses stand in a significant semantic relationship, and giving a precise specification of each semantic relationship thus exemplified.

2. Discovering rules by which we can compute from the W7 transcription a reasonable approximation to the recognition of just which word-senses in W7 stand in one of these specified semantic relationships to a randomly chosen word-sense.

An approach to the first task that we have found most fruitful has been to examine both diachronic changes of sense (as in a single word's change of meaning), and synchronic processes extending the possible meanings of a word or morpheme (as in the extension of spot n. to spot v., and of these to spotty, spotless, etc.). With this approach we are not confined initially to the slow process of analyzing meaning-changes and extensions under the innumerable W7 entries, but are freed to immediately apprehend productive processes and to test them against our knowledge of them as native speakers of English.

The findings summarized below were obtained by following this approach; a detailed discussion of these findings is given in SP-2893 and SP-2896 (see "Documentation").

1. Sense shift, whether in terms of low-level features such as ±human, ±count, etc., or in terms of higher syntactic categories or features such as +N, ±ADJ, +V, etc., seems to occur through a universal process of two steps: deletion of an underlying component, and (syntactic and semantic) re-interpretation of a remaining component. This remaining component "absorbs" the features of the deleted one, and is said to have undergone a sense shift.

2. It seems that the affixally marked and the affixally unmarked shifts are substantially similar, in terms of the rules underlying them. They seem to supply underlying lexical structure in much the same fashion, though they may differ on the surface of the utterance.

3. Rules of sense shift, including those entailing or accompanying category shift, are transformations of a very specific kind. The concept of structural description for these rules is quite different from the one developed for well-known "syntactic" transformations. The rules are very specific in function, deriving either new "senses" of words from existing senses, or new lexical items from existing items.

4. The place of lexicon-building transformations in the grammar of the language appears to be in a special section of the lexicon. The ramifications of this concept for rules of the semantic-interpretation component of the grammar have been investigated, but the analysis is not yet complete.

A separate investigation, thus far documented only in the form of a working paper, has demonstrated the rule-bound nature of metaphoric changes, and has explored in detail noun-to-verb conversions, showing that each different sense of a given denominal verb has its own derivational history (semantic and syntactic), which is unique relative to the derivational histories of every other sense of that verb, though sometimes merely a continuation of another such history (e.g., sense 3a may include the derivational history of sense 2c, with continuations). There are rather few underlying structures and "transformations" required for reconstruction of these derivational histories; only 8 main structures, comprising about 15 subtypes, have been described, though others will surely be found. These discoveries promise a simpler and more precise account of that part of the lexicon comprising nouns and denominal verbs.

An additional benefit of the work on denominal verbs is that methods devised to formulate their derivational histories have illuminated the process by which two verbs such as identify and spot (which at an earlier time had no semantic relation) have become synonymous for at least one sense of each. We are thus able to isolate and describe operations and operators involved in the development of synonymous relations. Recent study of diachronic changes that have created--by "convergence"--such a semantic field as that centering on the notion of money or that of time, indicates that the rules of change operative in creating these fields heavily overlap with those operative in the noun-to-verb changes. (To be more precise, these fields are not "created;" rather, a set of previously unrelated words is moved into the field, as with fee, salary, money, cash, whose original semantic fields were, respectively, those of cattle, salt, divine admonishment, and chests.) This suggests again that the types of semantic relations and semantic changes are a manageably limited set.

Our approach to the second major research task mentioned above--that of computing word-sense relationships from the W7 transcript--has been to analyze defining formulas used in W7. In most cases, a defining formula can be viewed as an n-place relation which, if it contains a word functioning

syntactically as head of the defining phrase in which it occurs, includes as one of its arguments either the word-sense being defined (e.g., ': characterized by ----') or the corresponding sense of an expression of which the word in question is a major constituent (e.g., [a ----] ': one who ----'). The other argument(s) are indicated by the expression(s) filling the blank(s) in the formula.

Fortunately, many defining formulas can be recognized accurately by machine; some can be recognized even without syntactic information about the defining phrases in which they occur, and thus can be used as clues for analyzing the phrases syntactically. Although defining formulas tend to be semantically ambiguous, the meaning differences involved in their ambiguity are usually subtle rather than gross. Thus we are often able to predict by approximation the semantic relations among word- and phrase-senses occupying argument places in defining formulas.

The transcription of W7 onto paper tape was completed in early March, approximately on schedule; the transcription of the Merriam-Webster Pocket Dictionary (MPD) onto paper tape was completed three weeks later. Selective sampling of the flexowriter printout indicated that the error rate was well within the maximum stipulated in the transcription contract (99.5 percent, or not more than 1 wrongly transcribed character out of every 200). A three-part plan (described below) was followed for converting the information on these tapes into forms in which it could be used at the earliest possible date by researchers on the project without unduly delaying the development of our projected interactive processor for tracing lexical fields.

1. Routinization of Type Conversion and Correction. The transcription of W7 consists of 174 reels of paper tape, each of which contains 6 pages of the dictionary followed by a list of corrections which were found by proofreaders and punched on the tape. The paper tapes containing MPD have the same format. At first we thought we could set up an essentially automatic, two-stage operation for (1) transcribing the paper tape onto magnetic tape and (2) integrating the transcribers' corrections into the dictionary text. It proved necessary to add a third stage: reading each paper tape back through the flexowriter to ensure that all the holes were punched through the tape, thus eliminating a major source of illegal characters being written on the magnetic tapes. (This procedure incidentally rewinds the paper tapes, which has to be done in any case before they are transcribed onto magnetic tapes.) We then discovered that a fair amount of human intervention was required for integrating the transcribers' corrections, owing to mismatches between some corrections and the entries to which they were supposed to apply. For example, a character in either the entry name or the reference to it in the correction might have been mistranscribed onto magnetic tape as an illegal character; or a correction might have been entered out of strict alphabetic order (with respect to the other corrections) by the original keystroker, etc.

The programs used in the interactive correction process were designed so that secretaries could make the required interventions after a few hours of instruction. To make effective use of their time during periods when the time-sharing system is heavily loaded or temporarily inoperative, a program was written to obtain printouts of the corrected magnetic tapes in a form easy to proofread, using the 120-character print chain available on the 360 computer. Secretaries can then check the tape during slack periods for errors that cannot readily be detected under program control, e.g., omitted lines. Thus our procedure for converting and correcting the transcription tapes has become a four-stage operation, requiring no programmer intervention in any of the stages. It appears that each of the stages can be operated efficiently and economically at the rate of 20 paper tapes per week; we have reached this rate for the first two stages (i.e., first in processing order) and expect to reach it for the other two very shortly.

2. Reformatting the Corrected Tapes. Without being reformatted, the corrected tapes can be searched for specified continuous character strings by using EDTEXT. Useful printouts of occurrences of defining formulas have already been obtained in this manner from the first 30 pages of W7. In response to requests from researchers on the project for W7 data that cannot be obtained in this way, a program was recently written and checked out to reformat the corrected tapes so that QUEST* can be used to satisfy some of these requests.

3. Parsing Corrected Tape Entries. The entries on the corrected tape are being parsed to prepare for more sophisticated processing of W7. Several of the researchers' requests cannot be satisfied by either EDTEXT or QUEST. Most of them require parsing W7 entries into the bold-face entry word or phrase, the pronunciation, the etymology, etc. Programs are now being written to satisfy high-priority requests, using a complete parsing algorithm for W7 entries that has already been specified in Backus-Naur Form (BNF).

   In preparation for satisfying those requests that require syntactic analyses of the defining phrases in W7, we have worked out a two-stage special-purpose analyzer in consultation with Susumu Kuno. In the first stage, a set of inflectional and affixal rules are used to determine the possible parts of speech of each word-form occurring in W7 sense descriptions. The second stage of the analyzer is designed to operate in either an interactive or automatic mode. In the interactive mode, a set of 250 syntactic rules are used to compute

*QUEST is a text-retrieval system developed previously under ARPA sponsorship of the Synthex project at SDC.

"surface-structure descriptions" of defining phrases on the basis of the part-of-speech labels assigned during the first stage. In the automatic mode, the analyzer will only mark the head(s) of each defining phrase but will operate at a sufficiently high speed so that it can process very large samples of the dictionary.

2. Transformational Grammar Tester

This project is concerned with developing a system of computer programs that will assist linguists in building and validating transformational generative grammars. The system is designed for operation on a time-shared computer and seeks to make full use of the advantages of man-machine interaction in a situation that allows the user virtually immediate access to the computer.

Chomsky--in Aspects of a Theory of Syntax--defines a generative grammar as one that "attempts to characterize in the most neutral possible terms the knowledge of the language that provides the basis for actual use of language by a speaker-hearer." He further defines it as "a system of rules that in some explicit and well-defined way assigns structural descriptions to sentences."

The syntactic component of such a grammar specifies the well-formed strings of formatives (minimal syntactically functioning elements) in the language and assigns structure to them. Transformational grammars are built on the concept of the separation of two types of structure. Accordingly, the syntactic component of a transformational grammar is made up of two sets of rules: phrase structure rules, which generate a deep structure, and transformations, which determine the ultimate surface structure of a sentence. The deep structure is operated on by the semantic component of the grammar to determine meanings. The surface structure is interpreted by the phonological component.

The emphasis on explicitness is a distinct advantage of the generative grammars. However, it imposes a great burden on the linguist. For example, determining the applicable rules in the derivation of a particular sentence is an extremely tedious and time-consuming task, difficult to perform with accuracy. And, as the grammar becomes large (as the linguist attempts to account for more phenomena in the language he is describing), it becomes more difficult to discover all of the interrelations of the rules.

In constructing a grammar with the transformational grammar tester developed under this project, the linguist can ask the same questions and employ many of the same procedures that he would have used were the tester not available. With the tester, those functions that are amenable to programming are available for operation by the linguist using a teletype and display scope, from which he also receives computer output.

The most important tasks being performed by the tester center around the ability to execute grammar rules. The linguist can use the computer to determine the applicability of transformations and to execute them and have their results displayed quickly and accurately. Many ancillary functions are provided for specification and manipulation of rules and test structures. Combinations of these functions greatly aid the linguist in the very important job of determining the implications of changes in the rule set. For example, the linguist can save entire derivations of sentences that he considers correct. He may then insert a new rule, or change or delete an existing one, and then have the computer apply the rule set to the base structures of the saved sentences. The computer will then display any changes in their derivations.

By the end of March 1967, an experimental model of a Transformational Grammar Tester (TGT) was completed. The TGT was written in LISP 1.5 for operation on the AN/FSQ-32 computer. Work was then begun on an operational model, TGT 2, designed to provide greater speed and more convenience to nonprogrammer users. The new model is user-independent in that it allows each user to create files of his own rules, "trees," and derivations (which reside in memory or on disc during operation), and to permanently save these files on tape, which may be input and tested or altered by subsequent runs.

Since the use of CRT display scopes requires proximity to the computer, TGT 2 has been designed to accommodate remote users by allowing all displays to be printed on-line at the user's option (see Figures 4 and 5). This also benefits the scope user, who may want a hard copy of the CRT displays.

The experimental model of TGT performed transformations of the form specified by Dr. Peter Rosenbaum in the IBM Core Grammar. TGT 2 provides a more general and more powerful rule schema. The speed of the transformation interpreter thus far has been very encouraging since it is almost certain that the grammar of our principal user, the Air Force-UCLA English Syntax Project, will require more computation than previous grammars (e.g., The MITRE Grammar or the IBM Core Grammar) because of the relatively large subset of English to be accounted for and because some of the proposed conventions for the grammar will allow transformations to apply to a given tree in more than one way. Also, operation in a time-sharing system with many users makes speed essential. Work on the transformation interpreter is still in progress and it is anticipated that modifications and improvements will continue as the UCLA group expands its skeleton grammar, and as efficient and desirable conventions for rule interpretation evolve.

The functions accepted by the TGT 2 control program can be divided into four groups:

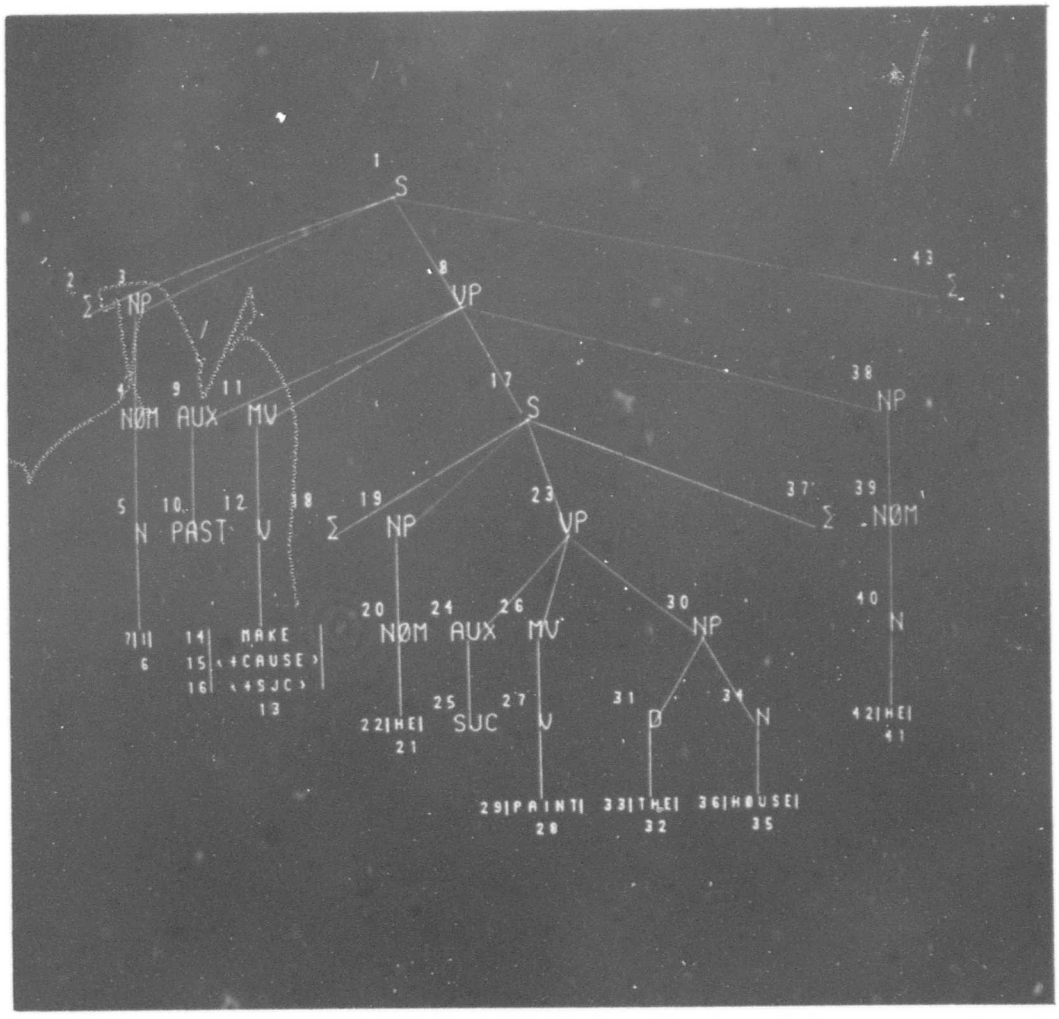

Figure 4. Tree Structure Generated by TGT Program on CRT Display

The tree shows the grammatical structure of the sentence "I made him paint the house."

```
(S 1)
  (# 2)
  (NP 3)
    (NOM 4)
      (N 5)
        (*CS* 6)[(I 7)]
  (VP 8)
    (AUX 9)
      (PAST 10)
    (MV 11)
      (V 12)
        (*CS* 13)[(MAKE 14)(<+CAUSE> 15)(<+SJC> 16)]
    (S 17)
      (# 18)
      (NP 19)
        (NOM 20)
          (*CS* 21)[(HE 22)]
      (VP 23)
        (AUX 24)
          (SJC 25)
        (MV 26)
          (V 27)
            (*CS* 28)[(PAINT 29)]
        (NP 30)
          (D 31)
            (*CS* 32)[(THE 33)]
          (N 34)
            (*CS* 35)[(HOUSE 36)]
      (# 37)
    (NP 38)
      (NOM 39)
        (N 40)
          (*CS* 41)[(HE 42)]
  (# 43)
```

Figure 5. Equivalent Tree Structure
Generated by TGT Program on Teletype

1. Tree creation and alteration: includes all of the primitives to print (or display) and number a tree, to erase substructures, and to substitute or add new structures in any of several ways. Many of these same functions are called internally to perform the structural change indicated by transformational rules.

2. Image positioning: these functions are used for trees too large to fit on the CRT. They allow the image of the tree to be left- or right-justified or to be centered on a particular node.

3. Inventory maintenance: inventories of rules, trees, and derivations may be created, named, and saved on disc or tape. A list of the items in the three inventories can be obtained. In addition, the order of the operation of rules can be specified.

4. Rule operation: the structural description of transformations may be selectively operated or applied as a set to a particular tree. Execution of the applicable transformations can be requested and a derivation (record of the rules applied) can be obtained.

3. Query Subset Studies

The purpose of this study has been to investigate principles which can be used by the designer of a data base query system to devise query subsets of natural languages that are best adapted to human use. Natural language query subsets have the advantage over artificial query languages of capitalizing on the inherent linguistic knowledge of the language learner to reduce his error rate in using the language.

The problem was to discover what aspects of a natural language could be used to predict human performance in regard to a given subset. Of the many variables that are--*a priori*--relevant, it was decided to concentrate on paraphrasability, i.e., the relative ease with which one expression can be substituted for another in the same context. In the use of a defined query language, a desirable goal is to have all the likely paraphrases of a given query acceptable to the system, and have only unlikely ones in the excluded set.

The methodology of the study was described in detail in the previous ARPA semiannual report.* In brief, the scheme was to have human subjects generate judgments of similarity among a set of English questions, and to multidimensionally analyze these judgments to determine the contribution to this similarity of various aspects of the grammar, particularly transformational rules. In a parallel effort, the preferences for certain grammatical

*SDC document TM-687/007/00, dated 31 December 1966, pp. 42-46.

forms would be measured. The measurements of similarity and preference would be combined to predict the error performance of several query subset languages that were to be developed.

One of the first steps taken in this project was to develop a grammar of English questions, which was, in fact, to be a grammar of normal English structures with an expanded grammatical treatment of question forms. Two grammars were developed. The first was an extension of the IBM Core Grammar (developed by Dr. Peter Rosenbaum of IBM), and the second was a more radical approach based primarily on the original ideas of Talmy Givón, a Research Associate on this project. These grammars are described in detail in TM-3565. It was decided to proceed with the more traditional grammar and (for the purposes of generating sentences) to collapse several of the many transformations into sets of transformational paradigms, in order to simplify the problem. Details of the generation of the corpus of English questions are outlined in TM-3566, along with a more complete listing of the material developed.

A study was performed by project personnel at UCLA to determine human preferences for various grammatical forms. In this study, graduate students majoring in marketing were required to query a census-type data base to obtain certain information required to solve some marketing problems posed to them. They were allowed to use unrestricted English for their queries. A corpus of some 4,000 questions was gathered in this manner. The plan was to have these questions parsed, using the model grammar, to obtain relative preference data in terms of syntactic descriptions of the questions.

PLANS

1. Lexicographic Studies

Our original plans called for implementation of an initial version of an interactive processor for tracing semantic fields within a year of the start date of the project. Two developments led us to extend this schedule by six months. On the one hand, the process of converting and correcting the paper tapes containing the W7 and MPD transcripts proved to be more time-consuming than we had anticipated. On the other hand, the quantity and diversity of requests for machine-derivable information about W7 from researchers on the project persuaded us to direct some of our efforts toward satisfying these real and pressing needs. These efforts would otherwise have gone into designing and implementing a processor for satisfying anticipated (but still potential) needs of linguists and philosophers for processing the transcripts. Unquestionably, the experience gained in satisfying these requests, as well as the insights into lexical structure that may be obtained from the information thus provided, will enable us to design and implement a more useful processor than if we had attempted to keep to the original schedule. In consultation with Jerome Feldman, we further decided to develop a special

dialect of META5* so that researchers can satisfy many of their own requests, giving the programmer more time to work on the processor for tracing semantic fields.

We are also investigating ways in which past and present conclusions regarding rules and principles operative in the lexicons of natural languages can be formulated. This will afford an understanding of exactly how the current and future results of the lexicographic studies at SDC fit into the theory of grammar as a whole, including both its semantic and syntactic aspects. It will also enable researchers to make appropriate distinctions among different kinds of lexical processes by providing a formal framework that is both narrow enough to enable one to state predictively the conventions and limitations on lexical processes and entries which have been discovered thus far, and--on the other hand--broad enough to allow for future formulations along this line.

2. Transformational Grammar Tester

Our efforts in the immediate future will involve the further development of programs for the operation of transformation rules. Then, in the sequence shown, we plan to do the following:

- Allow the user to determine if a particular tree could have been created by his phrase-structure rules.
- Allow an off-line mode of TGT, which would accept all inputs from tape and which would not be run in a time-sharing environment.
- Develop programs to aid in the creation, maintenance, and application of a large lexicon for the grammar.
- Transfer TGT to the IBM 360/67.

3. Query Subset Studies

Presently on hand, but unanalyzed, are the sets of queries for multidimensional scaling and the corpus of questions from the preference study. The steps remaining to complete the study are as follows: (1) have the sets of queries judged by humans for similarity; (2) make a multivariate analysis of these judgments; (3) parse and determine the relative frequency of the

*META5 is a compiler- or translator-writing program developed at SDC under ARPA sponsorship.

transformations in the preference data; and (4) construct the sample query languages and test the predictions made from the similarity data about the error rates in using these subsets.

At present, there are no plans to continue this work, as the project has been discontinued.

STAFF

R. F. Simmons, Head

Lexicographic Studies

J. C. Olney, Principal Investigator
D. L. Dwiggins
C. C. Revard
T. Givón, Research Associate
K. Machina, Research Associate
S. Trivus, Research Associate

Transformational Grammar Tester

D. L. Londe, Principal Investigator
W. J. Schoene

Query Subset Studies

R. L. Weiss, Principal Investigator
R. V. Katter
C. H. Kellogg
R. F. Simmons
T. Givón, Research Associate
R. Needleman, Research Associate
B. Mattison, Research Associate

DOCUMENTATION

Lexicographic Studies

Givón, T. Some noun-to-noun derivational affixes. SDC document SP-2893 (in press). 46 pp.

Describes a study of sense definitions of nouns as listed in Webster's Third International Dictionary. Particular emphasis was placed on the patterns of sense development induced on nouns by the suffixes -DOM, -HOOD, -SHIP, and -AGE. Senses were classified by their "syntactic features," upon which the sense shifts were defined. The paper suggests the following: (1) no formal distinction exists between the "syntactic" and "semantic" features or

processes; (2) sense development and derivation of new vocabulary items are the same process; (3) the semantic system of nouns in the English lexicon involves the hierarchy of nouns that underlie the meaning of other nouns, and the syntactic feature system employed in this study reflects this deeper system; (4) the lexicon of a language comprises sense listings, rules specifying the semantic structure of lexical items, and rules specifying the way by which various sense entries are related to each other or derived from each other.

Givón, T. Transformations of ellipsis, sense development, and rules of lexical derivation. SDC document SP-2896 (in press). 58 pp.

Presents the results of a study of the shifting of grammatical categories of lexical items (nouns, verbs, adjectives, etc.) without overt morphological affixation. These shifts were compared with derivational morphology and non-affixal sense development. The semantic processes underlying these operations were found to be identical; they were described by two steps: (1) deletion on the surface of the utterance, and (2) semantic reinterpretation of the remaining lexical material.

Query Subset Studies

Givón, T. Notes on the grammatical characterization of English queries. SDC document TM-3565 (in press). 26 pp.

Approaches the characterization of English questions transformationally along two alternative lines. First, a particular reference phrase structure grammar was adopted and question types enumerated in respect to this grammar. This enumeration did not constitute a "tight" grammar, but was--rather--translatable into one. This solution was then briefly evaluated with respect to several practical and linguistic problems involved. In particular, some questions relevant to the relation between a data base and the questions directed to it were investigated. An alternative approach was then suggested and briefly outlined. This approach consists of a much more abstract phrase structure grammar and will, therefore, require a much more complex transformational component, if the data base has English linguistic structure.

Needleman, R. and Mattison, R. The development of English queries for the query subset study. SDC document TM-3566 (in press). 10 pp.

Describes an attempt to lay the groundwork for determining query subsets of English that will be perceived by native speakers as "similar." Since the notion of "relatedness" of sentences is contained within transformational grammatical theory, this theory was used as a framework for the study. A model grammar that can characterize English sentences (which would then be submitted to English speakers for their "similarity judgments") is considered necessary for this work. A segment of such a grammar was developed, namely a phrase structure component and a limited number of transformations sufficient to generate a corpus of sentences for this study. The IBM Core Grammar was chosen for this task.

COMPUTER PROGRAM MANAGEMENT

Although some startling advances have recently been made in research on nonfunctional software (such as time-shared operating systems, data base management systems, and interactive programming aids), users have been unable to exploit the results of this work, since these advanced software systems are generally unavailable from computer manufacturers. Thus buyers are frequently disappointed with the nonfunctional software delivered by manufacturers. Part of this problem, however, stems from the fact that many buyers and users in government and industry are unable to specify computer programs and their development in ways aimed at assuring timely delivery of products with the desired performance characteristics. Furthermore, efforts to improve procurement practices have not been uniform nor have they been coordinated. Then too, ADP technology--both in research and in actual practice--continues to move ahead rapidly.

As a result, a study was undertaken at SDC to find uniform procurement procedures and to update outmoded procedures. This study was intended to contribute to these needed improvements by:

1. Suggesting changes to current definition and acquisition practices, in light of advanced technologies in nonfunctional computer programs.

2. Presenting guidelines for improved methods of specifying computer programs.

3. Identifying system design approaches and techniques that appear in nonfunctional software and that would make information-processing systems more versatile and thus extend their useful life.

PROGRESS

The report on the nonfunctional software study was completed and submitted to ESD on 26 June 1967. It contains a general statement of problems encountered by the buyers in acquiring this type of software, the difficulties encountered in specifying characteristics which such software must have, and the need for maintaining the greatest flexibility in system design so that users can change the software programs to meet their growing requirements. The study was restricted in its discussion to guidelines for acquisition of the following types of nonfunctional software:

- Executive systems
- Data base management systems
- Compilers
- Programmer aids

In addition to defining the performance characteristics of each of the above mentioned software, the report contains a discussion of the differences in nonfunctional software necessary because of the environment in which it functions, e.g., the difference between time-sharing and non-time-sharing software, or the differences in operating systems (e.g., load-and-go, batch-processing, multi-programming, etc.). The minimum hardware requirements for each type of nonfunctional software system are also specified.

An exhaustive study of this area would encompass a wide range of nonfunctional computer programs; it would involve examining the details of existing procurement methods and gathering extensive data on the policies, procedures, and plans for ADP development within WWMCCS. Within the constraints of practicality and the limitations of time, the scope of this study was restricted to interviews, discussions and literature surveys at SDC, MIT, MITRE, Auerbach, and Informatics.

PLANS

There are no definite plans for further continuation of this study.

STAFF

V. LaBolle, Project Leader
T. Fleishman
H. J. Ilger
R. N. Mathur
J. Kline

DOCUMENTATION

Ilger, H. J., Fleishman, T. and Mathur, R. N. Technical advice for procurement of nonfunctional computer programs: Preparation of a request for proposal for computer programs in multi-access systems. SDC document TM-3539/000/01. 29 June 1967. 56 pp.

Describes four types of nonfunctional (support) computer programs in terms of their major functions and the detailed performance characteristics corresponding to these functions. These descriptions supply inputs to Department of Defense (DOD) personnel to help them specify more detailed requirements for such computer programs in a Request For Proposal (RFP) issued to contractors. Detailed specifications in RFPs can improve understanding between DOD and its contractors and help assure the timely delivery of computer programs to satisfy real needs.

The document stresses the use of these computer programs--executives, compilers, data base management systems and programmer aids--in a multi-access automatic data processing center, particularly one which provides a time-sharing capability. This type of center is viewed as a possible prototype for automatic data processing in a fixed command center.

PROFESSIONAL ACTIVITIES

The following is a listing of the external presentations and publications of members of the staff for this reporting period. It includes contract-related items as well as other professional activities.

Barnett, J. A.

Chairman, Software Viewpoint Session, Hughes/RADC Conference on the Development of New Computer Organizations, Torrey Pines, California, 27-29 June 1967.

List processing in a paging environment. Presented at SHARE-28 Meeting, San Francisco, California, 15-16 February 1967.

Bernstein, M. I.

On-line character recognition. Presented at the ACM Professional Development Symposium on Computer Graphics, Atlantic City, New Jersey, 22 April 1967.

Book, E.

Invited participant, ACM Working Conference on On-Line Debugging, San Dimas, California, 9-11 January 1967.

Invited participant, Second AED (Extended Algol for Design) Technical Meeting, Massachusetts Institute of Technology, 25-27 January 1967.

Chairman, Los Angeles Chapter, SIGPLAN Working Group 1 on Syntax-Directed Compilers.

Book, E. and Schorre, D. V.

A simple compiler showing features of extended META. Presented at the SIGPLAN/PLANCOM Workshop on Compiler-Building Tools, Atlantic City, New Jersey, 14 April 1967. (Available as SDC document SP-2822.)

Erikson, W. J.

Simulation and gaming. In R. Archibald and R. Villoria (Eds.), Network-based management systems (PERT/CPM), New York: John Wiley & Sons.

Invited participant and panelist, Los Angeles Chapter of the ACM panel on Resource Utilization and Accounting in Time-Sharing Systems, Santa Monica, California, 4 January 1967.

Models for time-shared processing. Presented at the IEEE Symposium on Computers and Communications, Santa Monica, California, 19 January 1967.

Fleishman, T.
Development of cost-effectiveness relationships for computer programming. Resource Management Corp./SDC Seminar on Cost-Benefit Analysis, Santa Monica, California, 3-4 May 1967.

Kellogg, C. H.
Reviewer, Computing Reviews.

CONVERSE--a system for the on-line description and retrieval of structured data using natural language. Presented at the IFIP/FID Conference on Mechanized Information Storage, Retrieval and Dissemination, Rome, Italy, 14 June 1967. (Available as SDC document SP-2635.)

LaBolle, V.
Moderator, Computer Programming Management Panel, ACM Ninth Annual One-Day Technical Symposium, Los Angeles, California, 18 May 1967.

Londe, D. L.
Computers and natural language processing. Presented at California State College, Dominguez Hills, California, 18 May 1967.

Olney, J. C., Revard, C. C., and Ziff, P.
An interactive processor for a machine-usable version of Webster's Seventh New Collegiate Dictionary (W7) and the New Merriam-Webster Pocket Dictionary (MPD). The Finite String, March 1967, 4 (3), 1-2.

Revard, C. C.
An investigation of the structure of an English lexicon. Computers and the Humanities: A Newsletter, May 1967, 1 (5), 218-219.

A quantitative account of semantic changes in English words, 1000-1900 A.D. The Finite String, June 1967, 4 (6), 1-9.

The lecher, the legal eagle, and the papelard priest: Middle English confessional satires. In Donald E. Hayden (Ed.), His Firm Estate, the University of Tulsa Department of English Monograph Series, No. 2, 1967, 54-71.

Sackman, H.
An exploratory investigation of programmer performance under on-line and off-line conditions. Presented at the California State Psychological Association convention, San Diego, California, 28 January 1967.

Schaefer, M.
On the META5 language and system. Presented at the SIGPLAN/PLANCOM Workshop on Compiler-Building Tools, Atlantic City, New Jersey, 14 April 1967. (Available as SDC document SP-2823.)

Schwartz, J. I.
Analyzing time-sharing systems. Presented at the Second Congress on Electronic Information Handling, Testing and Evaluation, University of Pittsburgh, Pittsburgh, Pa., 14 April 1967.

Simmons, R. F.
Reviewer, Computing Reviews.

Automated language processing. In Carlos A. Cuadra (Ed.) Annual Review of Information Science and Technology, New York: John Wiley & Sons.

Natural language processing--progress and problems. Presented at the ACM Ninth Annual One-Day Technical Symposium, Los Angeles, California, 18 May 1967.

Linguistic inference for machine intelligence. Presented at the California Institute of Technology, Pasadena, California, 23 January 1967.

Verbal inference in question-answering systems. Presented at Leheigh University, Bethlehem, Pa., 16 February 1967.

Linguistic inference for machine intelligence. Presented at the University of Georgia, Athens, Ga., 16 January 1967.

Verbal inference for machine intelligence. Presented at the IBM Information and Linguistic Seminar, Yorktown Heights, New York, 17 February 1967.

Weissman, C.
LISP 1.5 primer. Belmont, California: Dickenson Publishing Company, Inc.

Invited participant and panelist, ACM Technical Symposium on Feasibility Problems in Time-Sharing, Los Angeles, California, 18 May 1967.

LISP 2: An introduction. Presented at the Rome/Utica Chapter of the ACM, Utica, New York, 14 May 1967.

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) System Development Corporation Santa Monica, California	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

Semiannual Technical Summary Report to the Director, Advanced Research Projects Agency for the Period 1 January 1967 to 30 June 1967

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

5. AUTHOR(S) (Last name, first name, initial)

6. REPORT DATE 30 June 1967	7a. TOTAL NO. OF PAGES 52	7b. NO. OF REFS
8a. CONTRACT OR GRANT NO. F1962867C0004, Information Processing Techniques b. PROJECT NO. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) TM-687/008/00 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	

10. AVAILABILITY/LIMITATION NOTICES

Distribution of this document is unlimited

11 SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY

13 ABSTRACT This report describes work done in the ARPA Information Processing Techniques Research and Laboratory Program at SDC from 1 January 1967 to 30 June 1967. Projects covered in this report include: Programming Language Development, Man-Machine Communication, Language Processing Research, and Computer Program Management.

DD FORM 1 JAN 64 1473

Unclassified

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Information Processing Computer Languages Computer Networks Man-Machine Communication Data Base Systems						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS), (S), (C), or (U)*

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.