CARNEGIE-MELLON UNIVERSITY
Computer Science Department

RESEARCH IN INFORMATION PROCESSING AND COMPUTER SCIENCE

FINAL TECHNICAL REPORT
Under AFOSR Contract #F44620-67-C-0058

Sponsored by

Advanced Research Projects Agency
ARPA Order No. 827

1. Research in Programming at Carnegie-Mellon University during 1968-1970

Three team programming efforts during the past two years have been the development of BLISS--a system building language on the PDP-10; LC^2--a conversational system on the IBM/360; and L*--a system building language on the PDP-10.

A number of additional individual programming language and system efforts have also been accomplished by graduate students as their doctoral dissertation work. These dissertations have appeared, or will appear shortly as ARPA reports.

Progress in programming is measured by results in the following problem areas:

(1) The definition of new language and the creation of a "running" system for it on one or more computers.

(2) The definition of new methodologies for building "running" programming systems for (preferably many) language(s).

(3) The study and formalization of one or more aspects of programming languages which are common to many specific languages.

During the contract period, work was performed in the above-mentioned areas.

(1) Progress in language definition

New languages are defined both to unify linguistic experience acquired with a diversity of languages and to create a base for new patterns of communication between man and computer. Of course, there is always a tinge of the former motive in an activity of the latter kind. Most of the language definition work at Carnegie during the contract period was, however, focussed on the latter goal. Little effort was made in defining new "total" languages in the Algol, LISP2, FORMULA ALGOL, PL/I, Algol 68 style.

(a) LC^2 (Language for Conversational Computing) was defined and created on the IBM/360-67. Here the goal was to exploit the execution methodology of time sharing in the execution of programs written in a problem solving language. The thesis--the mode in which programs written in a language L are executed strongly influences the language L--guided the definition of LCC as a language for conversational computing in a time-

sharing environment. A source language version of the system is available on magnetic tape. An ARPA report on LCC will be issued shortly and two brief papers about it have been published:

1. J. G. Mitchell, A. J. Perlis and H. Van Zoeren, "LC^2, a Language for Conversational Computing, in Interactive Systems for Experimental Applied Mathematics," edited by Reinfelds and Klerer, Academic Press, 1968.

2. J. G. Mitchell, "LC^2", Computer Science Research Review, 1969, Computer Science Department, Carnegie-Mellon University.

(b) BLISS was defined and created on the PDP-10. The increasing complexity of languages and systems has made it mandatory to improve the tools with which they are made. BLISS is a language meant to satisfy the programmer's need for creating efficient machine code without using machine assembly language. As such it provides the syntactic control elegance of Algol for manipulation of machine data structures (here the PDP-10) in such a way that its compiled programs are tightly organized and efficiently coded machine language programs. The test of such a system building language is, of course, the efficient systems constructible with it. Thus:

1. The BLISS compiler has been built in BLISS.
2. A WATFOR compiler has been built in BLISS.
3. An APL system is being built in BLISS.

Interestingly enough the WATFOR compiler was written by one man in about three months, though measurements of other efficiency criteria have not yet been obtained.

An ARPA report on BLISS has been produced. The BLISS system operates under the the PDP-10 time-sharing monitor.

(c) L* is a system building language also created on the PDP-10. The systems for which L* is intended are the large, dynamically organized, artificial intelligence systems which are generally built by one man. The design motifs here are simplicity and boot-strapping. The system initially provided to the user is made from about 1000 machine words and all else follows from that by an individually organized and

controlled boot-strapping process. Not only may systems be expanded, extended and grown, but they may be reorganized, compressed and distilled to reflect the influences of changing goals, improved techniques and better understanding. It is, by the way, the capability of compression and distillation that has been missing from almost all other extensible language systems.

The system is currently under test by a controlled group of users.

(2) Progress in system building methodology

Two of the language efforts (BLISS and L*) certainly contribute to system methodologies. An additional effort in system building methodology by a graduate student (Mr. Rudolph Krutar) is much more language independent. Called "conversational system programming," this methodology envisages systems as being constructed from "off-the-shelf" program modules by plugging them together as co-routines. Each of these modules has a number of ports for input and output of data and the modules are linked through their ports. Constructing a system is like wiring a patchboard in an analog computer. Once wired, pushing the "start button" causes the entire synchronized system to execute.

While it has been a long time goal in software to develop an "off-the-shelf" programming methodology, it has thus far been more of a dream than a reality and this work promises to bring that dream very close to fruition. The system was originally intended as a tool for development of an extensible formula manipulation language, but now appears more valuable as a general methodology than for the specific purpose for which it was originally intended.

(3) Programming languages can be characterized by the variability they make available to the programmer. Several pieces of research have been completed by graduate students which have focussed on the issue of variability and its converse--constancy.

David Fisher in his thesis, "Control Structures for Programming Languages," developed tools--and a carrier language--for describing control in programming languages. Whereas control had been taken as fixed in most

languages, Fisher has revealed how the programmer can fashion control that is most apt for the task being programmed. This thesis has appeared as an ARPA report (AD708511).

Jim Mitchell in his thesis, "The Design and Construction of Flexible Efficient Interactive Programming Systems," analyzes how interpretive conversational systems can be modified to provide compilation--with the resultant increase in running efficiency--for those parts of a program, e.g., in LCC, which are slowly changing, and still keep the programmer unaware of which parts are compiled; and even where compiled, the program is modifiable in the same flexible ways as when it was interpreted. Mitchell also treats methods of bootstrapping such systems into existence. His thesis has been released as an ARPA report. (No AD number has as yet been assigned.)

Gary Lindstrom in his thesis, "Variability in Language Processors," has considered the general question of variability in language processors:

(a) Freedom of base language selection: it should be possible to apply this system to a wide range of existing processors.

(b) Program representation flexibility: several interpretive execution levels should be available for the source language program.

(c) Incremental program structure: it is desirable for the program to exist as a variable collection of independently compiled segments, so as to permit mid-execution program recomposition.

(d) Dynamic program organization: the textual organization of the source program should be selectively replaceable by an organization based on execution sequence.

(e) Program execution ramification: it should be possible to generalize the concept of program execution to include several interacting but independent subexecutions.

(f) Programmer integration into execution: the relationship between the program and the programmer should be a symmetric one, in which neither possesses non-delegatable control over the execution.

He has developed a methodology for constructing highly variable language processors which goes far beyond what is currently available in systems. His thesis is to be released as an ARPA report.

Another aspect of programming languages is the correctness of programs written in them. A natural goal of computer science research is to develop a program for proving the correctness of sets of programs. The first such effort is described in the thesis, "A Program Verifier," of James King. While the language which can be treated is a simple one, the techniques used are applicable to real languages. A major next step will be the inclusion of functions in programs whose correctness is being proved. This thesis has been released as an ARPA report (AD599248).

The thesis, "Constructing Programs Automatically Using Theorem Proving," of Richard Waldinger deals with the creation of a program which will write programs from a problem statement--another major goal in Computer Science. This thesis has been released as an ARPA report (AD597041).

2. Research in Artificial Intelligence at CMU during 1968-70.

Progress in the field of artificial intelligence tends to center around the construction of total systems which demonstrate new sets of capabilities. Almost all such demonstrations fit one of four broad types:

(1) Competence in a new task domain

(2) High performance, usually on a narrow task

(3) Competence over a wide area -- generality

(4) Competence in a task of applied interest

Much rarer are attempts to evaluate particular reasoning mechanisms or attempts to put forth a theory for some class of mechanisms or some task domain. The last type on the list is a relatively recent addition, if we discount the close relationship between artificial intelligence and cognitive psychology, which has existed from the beginning.

The work at CMU during the last two years has members in all four of the above categories.

New Task Domains

The area of spatial design has been of interest at CMU for some time, primarily because of the presence of Charles Eastman, an architect, who held a joint position on the Computer Science faculty until last year. Eastman has attempted to understand how problems in design are specified by studying human design behavior (Eastman, 1968). The main technical issue is how to represent space inside the computer so that it can be manipulated with anything like the flexibility that humans use. One system has been developed by John Grason (1970). Another is nearing completion (Pfeffekorn). The latter is aimed at the problem of designing the layout for computer facilities (i.e., the locations of central processor, tapes, passageways, etc.).

A second task domain is that of programming. It is strange, to say the least, that the field of artificial intelligence has rather completely neglected the task of programming. A key issue has been that the program only provides a representation of the solution, and not of the problem. How is the goal to be stated for a programming task? Several answers have now been proposed. One of them, by Bob Floyd (1967), is embedded in an idea called the verifying compiler. This is a program that takes another program as input and attempts to prove that the program indeed does what it should. A verifying compiler system was constructed by King (1969) as his thesis, which realizes these ideas in a very impressive way. A related notion is to view the problem of programming as the task of proving a related theorem in a logical system. Given this proof, it should then be possible to extract out the elements for the program itself. This idea was realized in a thesis by Waldinger (1969). Both the efforts by King and by Waldinger represent important connections between artificial intelligence and the theory of programming (as represented, for example, by the work of Manna, discussed elsewhere in this report).

High Performance

The most important effort here is the research on speech recognition by Professor Reddy. He joined CMU in the Fall, 1969, from Stanford,

where he had an active program in speech recognition systems. He has spent the year building up his laboratory in connection with the new PDP-10 facility. Thus, there is little research to report. Two graduate students from Stanford accompanied him here and are doing their thesis work on aspects of speech recognition. The new system being built by Reddy here represents a third complete iteration in terms of program organization. The system built at Stanford showed impressive recognition capabilities, so that it is appropriate to consider this effort as a long range attempt to construct a high performance speech recognition system.

Chess has a well established position as a key problem in artificial intelligence. Enough people have worked on it, and enough discussion on the problems have occurred in the literature to make it continually fruitful With the efforts of Greenblatt at MIT, chess programs have reached a level of good club play, and it no longer is of interest just to put together a chess program. One is strictly aiming for high performance. We now have a chess program in this class, developed by Hans Berliner (a chess player of substantial caliber). It recently played in a chess tournament in Detroit, and in its present form it appears not to be quite as good as the present Greenblatt program. We expect to continue research into its structure and performance.

Generality

Work on the General Problem Solver (Ernst and Newell, 1969) terminated in 1968. However, since then there have been two efforts in the area of generality. One of these is a program developed by R. Fikes (1969, 1970) called REF-ARF, which takes as input an algebraic programming language (Algol-like) that has been augmented to permit variables so it can state problems (i.e., what values of the variables make the program terminate successfully?) This is a natural language for stating many kinds of problems. The problem solver takes the program as input and attempts to discover the correct values of the variables in the program. The program is quite successful and has been run on a wide variety of problems (Fikes, 1970, gives descriptions and statistics on 16 problems).

The second effort is a thesis by Donald Williams (1969) on analogy problems, such as occur in intelligence tests. The task for the program was not to do a particular kind of analogy problem, but to accept any of the variation that occur in the tests. (The program was limited to working on formal analogy problems, not involving the meanings of words.) The program was given the instructions to a particular test by being given the preliminary worked examples that occur in such tests to be sure that the test-taker understands the problem task. The program inducts the nature of the particular analogy problem from these examples and then goes on to apply this knowledge by doing a sequence of problems.

Applied Tasks

The stock of knowledge in artificial intelligence is beginning to approach the level at which applied tasks can be considered. The most outstanding example is the recent work at Stanford by Feigenbaum and his colleagues on Dendral, a program for inferring the structure of organic molecules from mass spectral data. Several efforts at CMU fall under this same rubric, though all of them are still in their early phases. They could all have been considered under the heading of "new task domains," but it seems more revealing to consider them as examples of attempts to move artificial intelligence in an applied direction.

Work is in progress on a program to design operating systems (Freeman, 1969). Work is also under way by Moore and Newell to construct a system that (in some sense understands the domain of artificial intelligence at the level of understanding the structure, function and performance of the various programs that make up the field. The applied motivation here is one of teaching and posing problems to students who wish to learn about artificial intelligence. There are severe conceptual difficulties in accomplishing this aim--i.e., in building understanding-programs for complex domains of knowledge. At least one other effort at CMU is attempting to shed light on this: work by W. Mann on what is required to understand the domain of sorting algorithms. A third applied effort is under way by D. Waterman and A. Newell to build a system for analyzing human problem solving data. This program will have to have considerable inferential powers, so that it is in fact an exercise in applied artificial

intelligence.

All of the above tasks are ambitious, and require a substantial phase of basic research for their successful accomplishment. They are applied efforts primarily in accepting a task domain that has applied interest.

Other Strands

Organizing the efforts in artificial intelligence in the four broad categories used above leaves out two important research efforts. One of these is the work on predicate calculus theorem proving, which is an active domain of artificial intelligence. Two faculty members have contributed heavily to this area (Loveland, 1969; Andrews, 1969). The other is the work in cognitive psychology at CMU, which depends heavily on the information processing concepts that have emerged from artificial intelligence. Most of this work is supported by other funds, although the work by Waterman and Newell cited above is a partial exception. However, the efforts work hand in hand. For instance, there is work on the perceptual and memorial organization of chess players, which is yielding important clues about how to organize chess programs. Also, a major effort during the last two years has been the preparation of a book which describes an information processing theory of how humans solve problems and is heavily dependent on work done at CMU over the last ten years (Newell and Simon, in press).

3. Systems

The analysis and design of systems--in their totality--is an increasingly important activity in computer science. In the thesis "Design and Behavior of TSS/8: A PDP-8 Based Time Sharing System," Ad Van de Goor produced a detailed analysis of a system he designed which serves as an admirable model for system designers. This thesis has been released as an ARPA report (AD707867)

In the thesis "The Description, Simulation, and Automatic Implementation of Digital Computer Processors," John Darringer developed a language for describing the behavior of digital computer processors irrespective of their eventual implementation. The programs so written can be translated into hardware specification for actual implementation. This thesis has been released as an ARPA report (AD700144).

REFERENCES

Andrews, Peter, "On Simplifying the Matrix of a Wff," Journal of Symbolic Logic, 33, (2), 180-192, (June, 1968).

Andrews, Peter, "Resolution with Merging," Journal of the Association for Computing Machinery, 15, (3), 367-381, (July, 1968).

Eastman, C. M., "Cognitive Processes and Ill Defined Problems: A Case Study from Designs," Proceedings of International Joint Conference on Artificial Intelligence, edited by D. E. Walker and L. M. Norton, The Mitre Corporation, (1969).

Ernst, G. W. and A. Newell, GPS: A Case Study in Generality and Problem Solving," Academic Press, New York, (1969).

Fikes, Richard E. (Ph.D. Dissertation), Research Mathematician, Stanford Research Institute, "A Heuristic Program for Solving Problems Stated as Nondeterministic Procedures," 1969, (AD 688604).

Floyd, Robert W., "Assigning Meanings to Programs," Proceedings of Symposia in Applied Mathematics, Volume 19: Mathematical Aspects of Computer Science, edited by J. T. Schwartz, American Mathematical Society, Providence, Rhode Island, 19-32 (1967).

Grason, John, (Ph.D. Dissertation), "Methods for the Computer-Implemented Solution of a Class of 'Floor Plan' Design Problems."

King, James Cornelius, (Ph.D. Dissertation), Research Staff, T. J. Watson Research Center, IBM Corporation, "A Program Verifier," (AD 699248)

Loveland, Donald W., "A Simplified Format for the Model Elimination Theorem-Proving Procedure," Journal of the Association for Computing Machinery, 16, 349-363, (1969).

Loveland, Donald W., "Theorem-Provers Combining Model Elimination and Resolution," Machine Intelligence, edited by Meltzer and Michie, Edinburgh University Press, 4, 73-86, (1969).

Loveland, Donald W., "Mechanical Theorem-Proving by Model Elimination," Journal of the Association for Computing Machinery, 15, (2) 236-251, (April, 1968).

Waldinger, Richard J. (Ph.D. Dissertation), Research Mathematician, Stanford Research Institute, "Constructing Programs Automatically Using Theorem Proving," 1969, (Ad 697041).

Williams, Donald S. (Ph.D. Dissertation), Systems Specialist, RCA Corporation, "Computer Program Organization Induced by Problem Example," 1969, (Ad 688242).

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)

Carnegie-Mellon University
Department of Computer Science
Pittsburgh, Pennsylvania 15213

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

RESEARCH IN INFORMATION PROCESSING AND COMPUTER SCIENCE

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

Scientific Final

5. AUTHOR(S) (First name, middle initial, last name)

Advanced Research Projects Agency

6. REPORT DATE

August 1970

7a. TOTAL NO. OF PAGES

10

7b. NO. OF REFS

13

8a. CONTRACT OR GRANT NO.

F44620-67-C-0058

b. PROJECT NO.

9718

c. 61102F

d. 681304

9a. ORIGINATOR'S REPORT NUMBER(S)

ARPA Order No. 827

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

AFOSR 70-2295 TR

10. DISTRIBUTION STATEMENT

1. This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES

TECH, OTHER

12. SPONSORING MILITARY ACTIVITY

Air Force Office of Scientific Research (SRM)
1400 Wilson Boulevard
Arlington, Virginia 22209

13. ABSTRACT

This is the Final Scientific Research for the research in Programming at Carnegie-Mellon University during 1968-1970.

DD FORM 1 NOV 65 1473

UNCLASSIFIED
Security Classification