APPENDIX C

FILM REPORTS

1. Art Eisenson and Gary Feldman, "Ellis D. Kropotechev and Zeus, his Marvelous Time-Sharing System", 16mm black and white with sound, runs about 15 minutes, March, 1967.

   Abstract: This film documents the advantages of time-sharing over standard batch processing. Through the good offices of the Zeus time-sharing system on the PDP-1 computer, our hero, Ellis, is saved from a fate worse than death. Recommended for mature audiences only.

2. Gary Feldman, "Butterfinger", 16mm color with sound, runs 8 minutes, March, 1968.

   Abstract: Describes the state of the hand-eye system at the Artificial Intelligence Project in the fall of 1967. The PDP-6 computer getting visual information from a television camera and controlling an electrical-mechanical arm solves simple tasks involving stacking blocks. The techniques of recognizing the blocks and their positions as well as controlling the arm are briefly presented.

3. Raj Reddy, Dave Espar and Art Eisenson, "Hear Here", 16mm color with sound, runs about 15 minutes, March, 1969.

   Abstract: Describes the state of the speech recognition project as of Spring, 1969. A discussion of the problems of speech recognition is followed by two real time demonstrations of the current system. The first shows the computer learning to recognize phrases and second show how the hand-eye system may be controlled by voice commands. Commands as complicated as "Pick up the small block in the lower lefthand corner," are recognized and the tasks required are carried out by the computer controlled arm.

4. Gary Feldman and Donald Peiper, "Avoid", 16mm silent,color, runs about 5 minutes, March, 1969.

   Abstract: Reports on a computer program written by D. Peiper for his Ph.D. thesis. The problem is to move the computer controlled electrical-mechanical arm through a space filled with one or more known obstacles. The programuses heuristics for finding a safe path; the film demonstrates the arm as it moves through various cluttered environments with fairly good success.

APPENDIX D ABSTRACTS OF ARTIFICIAL INTELLIGENCE PROJECT MEMOS

1963

1. J. McCarthy, Predicate Calculus with "undefined" as a Truth-Value, March
The use of predicate calculus in the matehmatical theory of computation and the problems involved in interpreting their values.

**2. J. McCarthy, Situations, Actions, and Causal Laws, July
A formal theory is given concerning situtations, causality and the possibility and effects of actions is given. The theory is intended to be used by the Advice Taker, a computer program that is to decide what to do by reasoning. Some simple examples are given of descriptions of situations and deductions that certain goals can be achieved.

3. F. Safier, "The Mikado" as an Advice Taker Problem, July
The situtation of the Second Act of "The Mikado" is analyzed from the point of view of Advice Taker formalism. This indicates defects still present in the language.

4. H. Enea, Clock Function for LISP 1.5, August
This paper describes a clock function for LISP 1.5.

5. H. Enea and D. Wooldridge, Algebraic Simplification, August
Herein described are proposed and effected changes and additions to Steve Russell's Mark IV Simplify.

*6. D. Wooldridge, Non-Printing Compiler, August
A short program which redefines parts of the LISP 1.5 compiler and suppresses compiler printout (at user's option) is described.

**7. J. McCarthy, Programs with Common Sense, September
Interesting work is being done in programming computers to solve probelms which require a high degree of intelligence in humans. However, certain elementary verbal reasoning processes so simple that they can be carried out by any non-feeble-minded human have yet to be simulated by machine programs.
This paper will discuss programs to manipulate in a suitable formal language (most likely a part of the predicate calculus)

---

* Out of print.

** Out of print. Reprinted as "Programs with Common Sense" in M. Minsky (Ed.), Semantic Information Processing, MIT Press, Cambridge, 1968.

common instrumental statements. The basic program will draw immediate conclusions from a list of premises. These conclusions will be either declarative or imperative sentences. When an imperative sentence is deduced the program takes a corresponding action. These actions may include printing sentences, moving sentences on lists, and reinitiating the basic deduction process on these lists.
Facilities will be provided for communication with humans in the system via manual intervention and display devices connected to the computer.

*8. J. McCarthy, Storage Conventions in LISP 2., September
Storage conventions and a basic set of functions for LISP 2 are proposed. Since the memo was written, a way of supplementing the features of this system with the unique storage of list structure using a hash rule for computing the address in a separate free storage area for lists has been found.

*9. C. M. Williams, Computing Estimates for the Number of Bisections of an N x N Checkerboard for N Even, December
This memo gives empirical justification for the assumption that the number of bisections of an N x N (N even) checkerboard is approximately given by the binomial coefficient $\binom{A}{\frac{A}{2}}$ where 2A is the length of the average bisecting cut.

10. S. R. Russell, Improvements in LISP Debugging, December
Experience with writing large LISP programs and helping students learning LISP suggests that spectacular improvements can be made in this area. These improvements are partly an elimination of sloppy coding in LISP 1.5, but mostly an elaboration of DEFINE, the push down list backtrace, and the current tracing facility. Experience suggests that these improvements would reduce the number of computer runs to debug a program a third to a half.

11. D. Wooldridge Jr., An Algebraic Simplify Program in LISP, December
A program which performs "obvious" (non-controversial)simplifying transformations on algebraic expressions (written in LISP prefix notation) is described. Cancellation of inverses and consolidation of sums and products are the basic accomplishments of the program; however, if the user desires to do so, he may request the program to perform special tasks, such as collect common factors from products in sums or expand products. Polynomials

* Out of print.

1963 (cont.)

are handled by routines which take advantage of the special form by polynomials; in particular, division (not cancellation) is always done in terms of polynomials. The program (run on the IBM 7090) is slightly faster than a human; however, the computer does not need to check its work by repeating the simplification. Although the program is usable - no bugs are known to exist - it is by no means a finished project. A rewriting of the simplify system is anticipated; this will eliminate much of the existing redundancy and other inefficiency, as well as implement an identity-recognizing scheme.

1964

*12. G. Feldman, Documentation of the MacMahon Squares Problem, January
An exposition of the MacMahon Squares problem together with some "theoretical" results on the nature of its solutions and a short discussion of an ALGOL program which finds all solutions are contained herein.

13. D. Wooldridge, The New LISP System (LISP 1.55), February
The new LISP system is described. Although differing only slightly it is thought to be an improvement on the old system.

14. J. McCarthy, Computer Control of a Machine for Exploring Mars, January
Landing a 5000 pound package on Mars that would spend a year looking for life and making other measurements has been proposed. We believe that this machine should be a stored program computer with sense and motor organs and that the machine should be mobile. We discuss the following points. 1. Advantages of a computer controlled system. 2. What the computer should be like. 3. What we can feasibly program the machine to do given the present state of work on artificial intelligence. 4. A plan for carrying out research in computer controlled experiments that will make the Mars machine as effective as possible.

15. M. Finkelstein and F. Safier, Axiomatization and Implementation, June
An example of a typical Advice-Taker axiomatization of a situation is given, and the situation is programmed in LISP as an indication of how the Advice-Taker could be expected to react. The situation chosen is the play of a hand of bridge.

16. J. McCarthy, A Tough Nut for Proof Procedures, July
It is well known to be impossible to tile with dominoes a checkerboard with two opposite corners deleted. This fact is readily stated in the first order predicate calculus, but the usual proof which involves a parity and counting argument does not readily translate into predicate calculus. We conjecture that this problem will be very difficult for programmed proof procedures.

17. J. McCarthy, Formal Description of the Game of Pang-Ke, July
The game of Pang-Ke is formulated in a first-order-logic in order to provide grist for the Advice-Taker Mill. The memo does not explain all the terms used.

* Out of print.

1964 (cont.)

*18. J. Hext, An Expression input Routine for LISP, July
The expression input routine is a LISP function, Mathread [ ] with associated definitions, which reads in expressions such as (A+3 - F(X,Y,Z)). Its result is an equivalent S-expression. The syntax of allowable expressions is given, but (unlike ALGOL's) it does not define the precedence of the operators; nor does the program carry out any explicit syntax analysis. Instead, the program parses the expression according to a set of numerical precedence values, and reports if it finds any symbol out of context.

19. J. Hext, Programming Languages and Translation, August
A notation is suggested for defining the syntax of a language in abstract form, specifying only its semantic constituents. A simple language is presented in this form and its semantic definition given in terms of these constituents. Methods are then developed for translating this language, first into a LISP format and from there to machine code, and for proving that the translation is correct.

20. R. Reddy, Source Language Optimization of For-Loops, August
Program execution time can be reduced, by a considerable amount, by optimizing the 'For-loops' of Algol Programs. By judicious use of index-registers and by evaluating all the sub-expressions whose values are not altered within the 'For loop', such optimization can be achieved.
In this project we develop an algorithm to optimize Algol Programs in List-structure form and generate a new source language program, which contains the "desired contents in the index registers" as a part of the For-clause of the For-statement and additional statements for evaluating the same expressions outside the 'For-loop'. This optimization is performed only for the innermost 'For-loops'.
The program is written entirely in LISP. Arrays may have any number of subscripts. Further array declarations may have variable dimensions. (Dynamic allocation of storage.)
The program does not try to optimize arithmetic expressions. (This has already been extensively investigated.)

21. R. W. Mitchell, LISP 2 Specifications Proposal, August
Specifications for a LISP 2 system are proposed. The source language is basically ALGOL 60 extended to include list processing, input/output and language extension facilities. The system would be implemented with a source language translator and optimizer, the output of which could be processed by either an interpreter or a compiler. The implementation is

* Out of print.

specified for a single address computer with particular reference to an IBM 7090 where necessary.
Expected efficiency of the system for list processing is significantly greater than the LISP 1.5 interpreter and also somewhat better than the LISP 1.5 compiler. For execution of numeric algorithms the system should be comparable to many current "algebraic" compilers.
Some familiarity with LISP 1.5, ALGOL and the IBM 7090 is assumed.

22. R. Russell, Kalah - The Game and the Program, September
A description of Kalah and the Kalah program, including subroutine descriptions and operating instructions.

23. R. Russell, Improvements to the Kalah Program, September
Recent improvements to the Kalah program are listed, and a proposal for speeding up the program by a factor of three is discussed.

24. J. McCarthy, A Formal Description of a Subset of Algol, September
We describe Microalgol, a trivial subset of Algol, by means of an interpreter. The notions of abstract syntax and of "state of the computation" permit a compact description of both syntax and semantics. We advocate an extension of this technique as a general way of describing programming language.

25. R. Mansfield, A Formal System of Computation, September
We discuss a tentative axiomatization for a formal system of computation and within this system we prove certain propositions about the convergence of recursive definitions proposed by J. McCarthy.

26. R. Reddy, Experiments on Automatic Speech Recognition by a Digital Computer, October
Speech sounds have in the past been investigated with the aid of spectrographs, vo-coders and other analog devices. With the availability of digital computers with improved i-o devices such as Cathode Ray tubes and analog digital converters, it has recently become practicable to employ this powerful tool in the analysis of speech sounds.
Some papers have appeared in the recent literature reporting the use of computers in the determination of the fundamental frequency and for vowel recognition. This paper discusses the details and results of a preliminary investigation conducted at Stanford. It includes various aspects of speech sounds such as waveforms of vowels and consonants; determination of a fundamental of the wave; Fourier (spectral) analysis of the sound waves formant determination, simple vowel recognition

1964 (cont.)

algorithm and synthesis of sounds. All were obtained by the use of a digital computer.

1965

27. J. McCarthy, A Proof-Checker for Predicate Calculus, March
A program that checks proofs in J. A. Robinson's formulation of predicate calculus has been programmed in LISP 1.5. The program is available in CTSS at Project MAC and is also available as a card deck. The program is used for class exercises at Stanford.

28. J. McCarthy, Problems in the Theory of Computation, March
The purpose of this paper is to identify and discuss a number of theoretical problems whose solutions seem feasible and likely to advance the practical art of computation. The problems that will be discussed include the following:
1. Semantics of programming languages. What do the strings of symbols representing computer programs, statements, declarations, labels, etc., denote? How can the semantics of programming languages be described formally?
2. Data spaces. What are the spaces of data on which computer programs act and how are they built up from simpler spaces?
3. How can time dependent and simultaneous processes be described?
4. Speed of computation. What can be said about how much computation is required to carry out certain processes?
5. Storage of information. How can information be stored so that items identical or similar to a given item can be retrieved?
6. Syntax directed computation. What is the appropriate domain for computations described by productions or other data format recognizers?
7. What are the appropriate formalisms for writing proofs that computer programs are equivalent?
8. In view of Gödel's theorem that tells us that any formal theory of computation must be incomplete, what is a reasonable formal system that will enable us to prove that programs terminate in practical cases?

29. C. M. Williams, Isolation of Important Features of a Multi-toned Picture, January
A roughly successful attempt is made to reduce a multi-toned picture to a two-toned (line drawing) representation capable of being recognized by a human being.

30. E. Feigenbaum and R. W. Watson, An Initial Problem Statement for a Machine Induction Research Project, April
A brief description is given of a research project presently getting under way. This project will study induction by machine, using organic chemistry as a task area. Topics for graduate student research related to the problem is listed.

1965 (cont.)

31. J. McCarthy, Plans for the Stanford Artificial Intelligence Project, April
The following is an excerpt from a proposal to ARPA and gives some of the project plans for the near future.

32. H. Ratchford, The 138 Analog Digital Converter, May
A discussion of the programming and hardware characteristics of the analog to digital converter on the PDP-1 is given; several sample programs are also presented.

33. B. Huberman, The Advice Taker and GPS, June
Using the formalism of the Newell-Shaw-Simon General Problem Solver to solve problems expressed in McCarthy's Advice Taker formalism is discussed. Some revisions of the formalism of can and cause described in AI Memo No. 2 are proposed.

34. P. Carah, A Television Camera Interface for the PDP-1, June
This paper is a discussion of several methods for the connection of a television camera to the PDP-1 computer. Three of these methods are discussed in detail and have in common that only a 36 bit portion of any horizontal scanning line may be read and this information is read directly into the working registers of the computer. The fourth involves a data channel to read information directly into the core memory of the computer, and is mentioned only in passing. The major concepts and some of the details of these methods are due to Marvin Minsky.

*35. F. Safier, Simple Simon, June
SIMPLE SIMON is a program which solves the problem of finding an object satisfying a predicate from a list of facts. It operates by backware chaining. The rules of procedure and heuristics are discussed and the structure of the program is outlined.

36. J. Painter, Utilization of a TV Camera on the PDP-1, September
A description of the programming required to utilize the TV camera connected to the PDP-1 and of the initial collection of programs.

37. K. Korsvold, An On Line Algebraic Simplification Program, November
We describe an on-line program for algebraic simplification. The program is written in LISP 1.5 for the Q-32 computer at System Development Corporation in Santa Monica, California. The program has in its entirety been written and debugged from a teletype station at Stanford University.

* Out of print.

1965 (cont.)

K. Korsvold, Appendix B, to A.I. 37
This appendix contains the program written in m-expressions. The four functions ADDK, TIMESKL, *GSD and *RFD are not included since they are written in LAP.

1966

38. D. Waterman, A Filter for a Machine Induction System, January
This report contains current ideas about the Machine Induction Research Project, and attempts to more clearly define some of the problems involved. In particular, the on-line data acquisition problem, the filter, and the inductive inference problem associated with the filter are discussed in detail.

39. K. Pingle, A Program to Find Objects in a Picture, January
A program is described which traces around objects in a picture, using the picture scanner attached to the PDP-1 computer, and fits curves to the edges.

40. J. McCarthy and J. Painter, Correctness of a Compiler for Arithmetic Expressions, April
This is a preprint of a paper given at the Symposium of Mathematical Aspects of Computer Science of the American Mathematical Society held April 7 and 8, 1966. It contains a proof of the correctness of a compiler for arithmetic expressions.

*41. P. Abrams and D. Rode, A Proposal for a Proof-Checker for Certain Axiomatic Systems, May
A proposed design for a proof-checker to operate on many axiomatic domains is presented. Included are descriptions of the organization and operation of the program to be written for the PDP-6.

42. K. Pingle, A Proposal for a Visual Input Routine, June
Some comments are made on the characteristics believed desirable in the next eye for the Stanford Artificial Intelligence Project and a proposal is given for a program to input scenes using the eye.

43. R. Reddy, An Approach to Computer Speech Recognition by Direct Analysis of the Speech Wave, September
A system for obtaining a phonemic transcription from a connected speech sample entered into the computer by a microphone and an analog-to-digital converter is described. A feature-extraction program divides the speech utterance into segments approximately corresponding to phonemes, determine pitch periods of those segments where pitch analysis is appropriate, and computes a list of parameters for each segment. A classification program assigns a phoneme-group label (vowel-like segment, fricative-like segment, etc.) to each segment, determines whether a segment should be classified as a phoneme or whether it represents a phoneme boundary between two phonemes, and then assigns a phoneme label to each segment that is not rejected as being a phoneme boundary. About 30 utterances of one to two seconds

* Out of print.

1966 (cont.)

duration were analyzed using the above programs on an interconnected IBM 7090 - PDP1 system. Correct identification of many vowel and consonantal phonemes was achieved for a single speaker. The time for analysis of each utterance was about 40 times real time. The results were encouraging and point to a new direction in speech research.

44. J. Painter, Semantic Correctness of a Compiler for an Algol-like Language, Revised March, 1967

This is a semantic proof of the correctness of a compiler. The abstract syntax and semantic definition are given for the language Mickey, an extension of Micor-algol. The abstract syntax and semantics are given for a hypothetical one-register single-address computer with 14 operations. A compiler, using recursive descent, is defined. Formal definitions are also given for state vector, a and c functions, and correctness of a compiler. Using these definitions, the compiler is proven correct.

45. D. Kaplan, Some Completeness Results in the Mathematics Theory of Computation, October

A formal theory is described which incorporates the "assignment" function $a(i, k, \xi)$ and the "contents" function $c(i, \xi)$. The axioms of the theory are shown to comprise a complete and consistent set.

46. S. Persson, Some Sequence Extrapolating Programs: A Study of Representation and Modeling in Inquiring Systems, September.

The purpose of this thesis is to investigate the feasibility of designing mecahnized inquiring-systems for finding suitable representations of problems, i.e., to perform the "creative" task of finding analogies. Because at present a general solution to this problem does not seem to be within reach, the feasibility of mechanizing a particular representational inquirer is chosen as a reasonable first step towards an increased understanding of the general problem. It is indicated that by actually designing, programming and running a representational inquirer as a program for a digital computer, a severe test of its consistency and potential for future extensions can be performed.

*47. B.Buchanan, Logics of Scientific Discovery, December.

The concept of a logic of discovery is discussed from a philosophical point of view. Early chapters discuss the concept of discovery itself, some arguments which have been advanced against logics of discover, notably by N. R. Hanson, and

* Out of print. Available through University Microfilms, 300 N. Zeeb Road, P.O. Box 1346, Ann Arbor, Michigan 48106.

1966 (cont.)

S. E. Toulmin. While a logic of discovery is generally understood to be an algorithm for formulating hypotheses, other concepts have been suggested. Chapters V and VI explore two of these: (A) a set of criteria by which a hypotheses could be judged reasonable, and (B) a set of rational (but not necessarily effective) methods for formulating hypotheses.

1967

48. D. Kaplan, Correctness of a Compiler for Algol-like Programs, July
A compiling algorithm is given which maps a class of Algol-like programs into a class of machine language programs. The semantics, i.e., the effect of execution, of each class if specified, and recursion induction used to prove that program semantics is preserved under the mapping defined by the compiling algorithm.

49. G. Sutherland, DENDRAL - A Computer Program for Generating and Filtering Chemical Structures, February
A computer program has been written which can generate all the structural isomers of a chemical composition. The generated structures are inspected for forbidden substructures in order to eliminate structures which are chemically impossible from the output. In addition, the program contains heuristics for determining the most plausible structures, for utilizing supplementary data, and for interrogating the on-line user as to desired options and procedures. The program incorporates a memory so that past experiences are utilized in later work.

50. A. Hearn, Reduce Users' Manual, February
REDUCE is a program designed for general algebraic computations of interest to physicists and engineers. Its capabilities include:
1) expansion and ordering of rational functions of polynomials,
2) symbolic differentiation,
3) Substitutions in a wide variety of forms,
4) reduction of quotients of polynomials by cancellation of common factors,
5) calculation of symbolic determinants,
6) calculations of interest to high energy physicists including spin 1/2 and spin 1 algebra.

The program is written completely in the language LISP 1.5 and may therefore be run with little modification on any computer possessing a LISP 1.5 compiler or interpreter.

51. L. Earnest, Choosing an Eye for a Computer, April
In order for a computer to operate efficiently in an unstructured environment, it must have one or more manipulators (e.g., arms and hands) and a spatial sensor analogous to the human eye. Alternative sensor systems are compared here in their performance on certain simple tasks. Techniques for determining color, texture, and depth of surface elements are examined. Sensing elements considered include the photomultiplier, image dissector, image orthicon, vidicon, and SEC camera tube. Performance measures strongly favor a new (and undemonstrated) configuration that may be termed a laser jumping spot system.

1967 (cont.)

52. A. L. Samuel, Some Studies in Machine Learning Using the Game of Checkers II - Recent Progress, June

A new signature table technique is described together with an improved book learning procedure which is thought to be much superior to the linear polynomial method described earlier. Full use is made of the so called "alpha-beta" pruning and several forms of forward pruning to restrict the spread of the move tree and to permit the program to look ahead to a much greater depth than it otherwise could do. While still unable to outplay checker masters, the program's playing ability has been greatly improved. Some of these newer techniques should be applicable to problems of economic importance.

53. B. Weiher, The PDP-6 Proof Checker, June

A description if given for the use of a proof checker for propositional calculus. An example of its use as well as the M and S expressions for the proof checker are also included.

54. J. Lederberg and E. A. Feigenbaum, Mechanization of Inductive Inference in Organic Chemistry, August

A computer program for formulating hypotheses in the area of organic chemistry is described from two standpoints: artificial intelligence and organic chemistry. The Dendral Algorithm for uniquely representing and ordering chemical structures defines the hypothesis-space; but heuristic search through the space is necessary because of its size. Both the algorithm and the heuristics are described explicitly but without reference to the LISP code in which these mechanisms are programmed. Within the program some use has been made of man-machine interaction, pattern recognition, learning, and tree-pruning heuristics as well as chemical heuristics which allow the program to focus its attention on a subproblem to rank the hypotheses in order of plausibility. The current performance of the program is illustrated with selected examples of actual output showing both its algorithmic and heuristic aspects. In addition some of the more important planned modifications are discussed.

55. J. Feldman, First Thoughts of Grammatical Inference, August.

A number of issues relating to the problem of inferring a grammar are discussed. A strategy for grammatical inference is presented and its weaknesses and possible improvements are discussed. This is a working paper and should not be reproduced, quoted or believed without the author's permission.

1967 (cont.)

56. W. Wichman, Use of Optical Feedback in the Computer Control of an Arm, August.

This paper reports an experimental investigation of the application of visual feedback to a simple computer-controller block-stacking task. The system uses a vidicon camera to examine a table top containing two cubical blocks, generating a data structure which is analyzed to determine the position of one block. An electric arm picks up the block and removes it from the scene, then after the program locates the second block, places the first on top of the second. Finally, the alignment of the stack is improved by analysis of the relative position error as seen by the camera. Positions are determined throughout by perspective transformation of edges detected from a single viewpoint, using a support hypothesis to supply sufficient information on depth. The Appendices document a portion of the hardware used in the project.

57. A. C. Hearn, Reduce, A User-Oriented Interactive System for Algebraic Simplification, October

This paper describes in outline the structure and use of REDUCE, a program designed for large-scale algebraic computations of interest to applied mathematicians, physicists and engineers. The capabilities of the system include:

1) expansion, ordering and reduction of rational functions of polynomials,
2) symbolic differentiation,
3) substitutions for variables and expressions appearing in other expressions,
4) simplification of symbolic determinants and matrix expressions,
5) tensor and non-commutative algebraic calculations of interest to high energy physicists.

In addition to the operations of addition, subtraction, multiplication, division, numerical exponentiation and differentiation, it is possible for the user to add new operators and define rules for their simplification. Derivations of these operators may also be defined.

The program is written complete in the language of LISP 1.5 and is organized so as to minimize the effort required in transferring from one LISP system to another.

Some particular problems which have arisen in using REDUCE in a time-sharing environment are also discussed.

58. M. D. Callero, An Adaptive Command and Control System Utilizing Heuristic Learning Processes, December

The objectives of the research reported here are to develop an automated decision process for real time allocation of defense missiles to attacking ballistic missiles in general war and to demonstrate the effectiveness of applying heuristic learning to seek optimality in the process. The approach is to model and simulate a missile defense environment and generate a

1967 (cont.)

decision procedure featuring a self-modifying, heuristic decision function which improves its performance with experience. The goal of the decision process that chooses between the feasible allocations is to minimize the total effect of the attack, measured in cumulative loss of target value. The goal is pursued indirectly by considering the more general problem of maintaining a strong defense posture, the ability of the defense system to protect the targets from both current and future loss. Using a simulation and analysis, a set of calculable features are determined which effectively reflect the marginal deterioration of defense posture for each allocation in a time interval. A decision function, a linear polynomial of the features, is evaluated for each feasible allocation and the allocation having the smallest value is selected. A heuristic learning process is incorporated in the model to evaluate the performance of the decision process and adjust the decision function coefficients to encourage correct comparison of alternative allocations. Simulated attacks presenting typical defense situations were cycled against the decision procedure with the result that the decision function coefficients converged under the learning process and the decision process become increasingly effective.

1968

59. D. M. Kaplan, A Formal Theory Concerning the Equivalence of Algorithms, May

Axioms and rules of inference are given for the derivation of equivalence for algorithms. The theory is shown to be complete for certain subclasses of algorithms, and several applications of the theory are illustrated. This paper was originally presented at the Mathematical Theory of Computation Conference, IBM Yorktown Heights, November 27-30, 1967.

60. D. M. Kaplan, The Formal Theoretic Analysis of Strong Equivalence for Elemental Programs, June

The syntax and semantics is given for elemental programs, and the strong equivalence of these simple ALGOL-like flowcharts is shown to be undecidable. A formal theory is introduced for deriving statements of strong equivalence, and the completeness of this theory is obtained for various sub-cases. Several applications of the theory are discussed. Using a regular expression representation for elemental programs and an unorthodox semantics for these expressions, several strong equivalence detecting procedures are developed. This work was completed in essentially its present form March, 1968.

61. T. Ito, Notes of Theory of Computation and Pattern Recognition, May

This is a collection of some of the author's raw working notes during the period December 1965 - October 1967 besides the introduction. They have been privately or internally distributed for some time. Portions of this work have been accepted for publication; others are being developed for submission to journals. Some aspects and ideas have been referred to and used, sometimes without explicit references, and others are developed by other researchers and the author. Hence we have decided to publish this material as Computer Science Technical Report, although the author is planning to submit all of these works to some journals, adding several new results (not mentioned in this report), improving notations, definitions and style of presentation in some parts and reformulating completely in other parts. The author appreciates it very much of the researchers who use or refer to the results and ideas of this report communicate with him. The publication of this report was encouraged by Prof. George E. Forsythe and Prof. John McCarthy.

62. B. Buchanan and G. Sutherland, HEURISTIC DENDRAL: A Program for Generating Explanatory Hypotheses in Organic Chemistry, July

A computer program has been written which can formulate hypotheses from a given set of scientific data. The data consist

of the mass spectrum and the empirical formula of an organic chemical compound. The hypotheses which were produced describe molecular structures which are plausible explanations of the data. The hypotheses are generated systematically within the program's theory of chemical stability and within limiting constraints which are inferred from the data by heuristic rules. The program excludes hypotheses inconsistent with the data and lists its candidate explanatory hypotheses in order of decreasing plausibility. The computer program is heuristic in that it searches for plausible hypotheses in a small subset of the total hypothesis space according to heuristic rules learned from chemists.

63. D. M. Kaplan, Regular Expressions and the Equivalence of Programs, July

The strong equivalence of ALGOL-like programs is, in general, an undecidable property. Several mechanical procedures are discussed which nevertheless are useful in the detection of strong equivalence. These methods depend on a regular expression representatation of programs. An unorthodox semantics for these expressions is introduced which appreciably adds to the ability to detect strong equivalence. Several other methods of extending this ability are also discussed.

64. Z. Manna, Formalization of Properties of Programs, July

Given a program, an algorithm will be described for constructing an expression, such that the program is valid (i.e., terminates and yields the right answer) if and only if the expression is inconsistent. Similar result for the equivalence problem of programs is given. These results suggest a new approach for proving the validity and the equivalence of programs.

65. B. Huberman, A Program to Play Chess End Games, August

A program to play chess end games is described. The model used in the program is very close to the model assumed in chess books. Embedded in the model are two predicates, better and worse, which contain the heuristics of play, different for each end game. The definitions of better and worse were obtained by programmer translation from the chess books.

The program model is shown to be a good one for chess and games by the success achieved for three end games. Also the model enables us to prove that the program can reach checkmate from any starting position. Insights about translation from book problem solving methods into computer program heuristics are discussed; they are obtained by comparing the chess book methods with the definitions of better and worse, and by considering the difficulty encountered by the programmer when doing the translation.

1968 (cont.)

66. J. Feldman and P. Rovner, An Algol-Based Associative Language, August

A high-level programming language for large complex relational structures has been designed and implemented. The underlying relational data structure has been implemented using a hash-coding technique. The discussion includes a comparison with other work and examples of applications of the language. A version of this paper will appear in the communications of the ACM.

67. E. Feigenbaum, Artificial Intelligence: Themes in the Second Decade, August

In this survey of artificial Intelligence research, the substantive focus is heuristic programming, problem solving, and closely associated learning models. The focus in time is the period 1963-1968. Brief tours are made over a variety of topics: generality, integrated robots, game playing, theorem proving, semantic information processing, etc.
One program, which employs the heuristic search paradigm to generate explanatory hypotheses in the analysis of mass spectra of organic molecules, is described in some detail. The problem of representation for problem solving systems is discussed. Various centers of excellence in the artificial intelligence research area are mentioned. A bibliography of 76 references is given.

68. Z. Manna and A Pnueli, The Validity Problem of the 91-Function, August

Several methods for proving the weak and strong validity of algorithms are presented.
For proving the weak validity (i.e., correctness) we use satisfiability methods, while for proving the strong validity (i.e., termination and correctness) we use unsatisfiability methods.
Two types of algorithms are discussed: recursively defined functions and programs.
Among the methods we include known methods due to Floyd, Manna, and McCarthy. All the methods will be introduced quite informally by means of an example (the 91-function).

69. J. McCarthy, Project Technical Report, September.

Recent work of Stanford Artificial Intelligence Project is summarized in several areas:

Scientific Hypothesis Formation
Symbolic Computation
Hand-Eye Systems
Computer Recognition of Speech
Board Games
Other Projects

1968 (cont.)

70. A. C. Hearn, The Problem of Substitution, December
One of the most significant features of programs designed for non-numeric calculation is that the size of expressions manipulated, and hence the amount of storage necessary, changes continually during the execution of the program. It is therefore usually not possible for the user to know ahead of time just how much output his program will produce, or whether the calculation will in fact fail because of lack of available computer memory. The key to keeping both the size of intermediate expressions and output under control often lies in the manner in which substitutions for variables and expressions declared by the programmer are implemented by the system. In this paper various methods which have been developed to perform these substitutions in the author's own system REDUCE are discussed. A brief description of the REDUCE system is also given.

71. P. Vicens, Preprocessing for Speech Analysis, October
This paper describes a procedure, and its hardware implementation, for the extraction of significant parameters of speech. The process involves division of the speech spectrum into convenient frequency bands, and calculation of amplitude and zero-crossing parameters in each of these bands every 10 ms. In the software implementation, a smooth function divides the speech spectrum into two frequency bands (above and below 1000 Hz). In the hardware implementation, the spectrum is divided into three bands using bandpass filters (150-900 Hz, 900-2200 Hz, 2200-5000 Hz). Details of the design and implementation of the hardware device are given.

72. D. L. Pieper, The Kinematics of Manupulators Under Computer Control, October
The kinematics of manipulators is studied. A model is presented which allows for the systematic description of new and existing manipulators.
Six degree-of-freedom manipulators are studied. Several solutions to the problem of finding the manipulator configuration leading to a specified position and orientation are presented. Numerical as well as explicit solutions are given. The problem of positioning a multi-link digital arm is also discussed.
Given the solution to the position problem, as a set of heuristics is developed for moving a six degree-of-freedom manipulator from an initial position to a final position through a space containing obstacles. This results in a computer program shown to be able to direct a manipulator around obstacles.

1968 (cont.)

73. John McCarthy, Some Pilosophical Problems from the Standpoint of Artificial Intelligence, November

A computer program capable of acting intelligently in the world must have a general representation of the world in terms of which its inputs are interpreted. Designing such a program requires commitments about what knowledge is and how it is obtained. Thus some of the major traditional problems of philosophy arise in artificial intelligence.

More specifically, we want a computer program that decides what to do by inferring in a formal language that a certain strategy will achieve its assigned goal. This requires formalizing concepts of causality, ability, and knowledge. Such formalisms are also considered in philosophical logic.

The first part of the paper begins with a philosophical point of view that seems to arise naturally once we take seriously the idea of actually making an intelligent machine. We go on to the notions of metaphysically and epistemologically adequate representations of the world and then to an explanation of can, causes, and knows, in terms of a representation of the world by a system of interacting automata. A proposed resolution of the problem of freewill in a deterministic universe and of counterfactual conditional setences is presented.

The second part is mainly concerned with formalisms within which it can be proved that a strategy will achieve a goal. Concepts of situation, fluent, future operator, action, strategy, result of a strategy and knowledge are formalized. A method is given of constructing a sentence of first order logic which will be true in all models of certain axioms if and only if a certain strategy will achieve a certain goal.

The formalism of this paper represents an advance over (McCarthy 1963) and (Green 1968) in that it permits proof of the correctness of strategies that contain loops and strategies that involve the acquisition of knowledge, and it is also somewhat more concise.

The third part discusses open problems in extending the formalism of Part II.

The fourth part is a review of work in philosophical logic in relation to problems of artificial intelligence and discussion of previous efforts to program "general intelligence" from the point of view of this paper. This paper is based on a talk given to the 4th Machine Intelligence Workshop held at Edinburgh, August 12-21, 1968, and is a preprint of a paper to be published in 'Machine Intelligence 4' (Edinburgh University Press, 1969).

1968 (cont'd.)

*74. D. Waterman, Machine Learning of Heuristics,
The research reported here is concerned with devising machine-learning techniques which can be applied to the problem of automating the learning of heuristics.

75. R.C. Schank, A Notion of Linguistic Concept: A Prelude to Mechanical Translation
The conceptual dependency framework has been used as an automatic parser for natural language. Since the parser gives as output a conceptual network capable of expressing meaning in language-free terms it is possible to regard this as an interlingua. If an interlingua is actually available how might this interlingua be used in translation? The primary problem that one encounters is the definition of just what these concepts in the network are. A concept is defined as an abstraction in terms of percepts and the frequency of connection of other concepts. This definition is used to facilitate the understanding of some of the problems in paraphrasing and translation. The motivation for this abstract definition of linguistic concept is discussed in the context of its proposed use.
DESCRIPTORS: Computational Linguistics, Concepts Research, Computer Understanding.

76. R.C. Schank, A Conceptual Parser for Natural Language,
This paper describes an operable automatic parser for natural language. The parser is not concerned with producing the syntactic structure of an input sentence. Instead, it is a conceptual parser, concerned with determining the underlying meaning of the input. The output of the parser is a network of concepts explicating the conceptual relationships in a piece of discourse. The structure of this network is language-free; thus, sentences in different languages or paraphrases within the same language will parse into the same network. The theory behind this representation is outlined in this paper and the parsing algorithm is explained in some detail.
DESCRIPTORS: Computational Linguistics, Concepts, Linguistic Research, Computer Understanding.

*Out of print.

1969

77. J. D. Becker, The Modeling of Simple Analogic and Inductive Processes in a Semantic Memory System, January

In this paper we present a general data structure for a semantic memory, which is distinguished in that a notion of consequence (temporal, causal, logical, or behavioral, depending on interpretation) is a primitive of the data representation. The same item of data may at one time serve as a logical implication, and at another time as a "pattern/action" rule for behavior.

We give a definition of "analogy" between items of semantic information. Using the notions of consequence and analogy, we construct an inductive process in which general laws are formulated and verified on the basis of observations of individual cases. We illustrate in detail the atainment of the rule "Firemen wear red suspenders" by this process.

Finally, we discuss the relationship between analogy and induction, and their use in modeling aspects of "perception" and "understanding".

78. D. R. Reddy, On the Use of Environmental, Syntactic, and Probalistic Constraints in Vision and Speech, January

In this paper we consider both vision and speech in the hope that a unified treatment, illustrating the similarities, would lead to a better appreciation of the problems, and possibly programs which use the same superstructure. We postulate a general perceptual system and illustrate how various existing systems either avoid or ignore some of the difficult problems that must be considered by a general perceptual system. The purpose of this paper is to point out some of the unsolved problems, and to suggest some heuristics that reflect environmental, syntactic, and probabilistic constraints useful in visual and speech perception by machine. To make effective use of these heuristics, a program must provide for

1. An external representation of heuristics for ease of man-machine communication
2. An internal representation of heuristics for effective use by machine
3. A mechanism for the selection of appropriate heuristics for use in a given situation.

Machine perception of vision and speech, thus, provides a problem domain for testing the adequacy of the models of representation (McCarthy and Hayes), planning and heuristic selection (Minsky, Newell and Simon), and generalization learning (Samuel); a domain in which (perceptual) tasks are performed by people easily and without effort.

79. D. R. Reddy and R. B. Neely, Contextual Analysis of Phonemes of English, January

It is now well known that the acoustic characteristics of a