Nils Nilsson

Final Report

Covering the Period 29 January 1973 to 30 June 1973

PAJARO DUNES WORKSHOP ON AUTOMATIC PROBLEM SOLVING

By: NILS J. NILSSON

Prepared for:

MR. MARVIN DENICOFF
CODE 437
OFFICE OF NAVAL RESEARCH
800 NORTH QUINCY
ARLINGTON, VIRGINIA 22217

CONTRACT N00014-73-C-0245

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

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

Final Report

Covering the Period 29 January 1973 to 30 June 1973

July 1973

PAJARO DUNES WORKSHOP ON AUTOMATIC PROBLEM SOLVING

By: NILS J. NILSSON

Prepared for:

MR. MARVIN DENICOFF
CODE 437
OFFICE OF NAVAL RESEARCH
800 NORTH QUINCY
ARLINGTON, VIRGINIA 22217

CONTRACT N00014-73-C-0245

SRI Project 2527

Approved by:

B. RAPHAEL, *Director*
Artificial Intelligence Center

B. COX, *Executive Director*
Information Science and Engineering Division

ABSTRACT

This report describes the talks and discussions occurring at a Workshop on Automatic Problem Solving held at Pajaro Dunes, California, on May 14-16, 1973.

I INTRODUCTION

Stanford Research Institute recently organized an informal workshop on Automatic Problem Solving, under the sponsorship of the Information Systems Branch of the Office of Naval Research (Contract No. N00014-73-C-0245). The workshop took place at Pajaro Dunes, California on May 14-16, 1973. In this report we shall present a brief summary of the workshop proceedings.

Those who attended the workshop are world leaders in the field of automatic problem solving. Both this workshop and an earlier one held at Firbush Point, Loch Tay, Scotland in March 1972 resulted in quite valuable interchange.

Automatic problem solving is a key area of research in Artificial Intelligence (A.I.). It is anticipated that advances in this field will lead to computer systems with much more flexibility and problem-solving power. Developments in this field have an impact on systems for information retrieval, language understanding, mathematical theorem proving, speech recognition and understanding, and command and control. The several projects on robot systems, for example, draw directly on research in automatic problem solving.

II PARTICIPANTS

The participants and their affiliations are listed below:

Bruce Anderson
Artificial Intelligence Laboratory
Stanford University

Robert Balzer
Information Sciences Institute
University of Southern California

Harry Barrow
School of Artificial Intelligence
University of Edinburgh

Robert Boyer
School of Artificial Intelligence
University of Edinburgh

Ted Elcock
University of Western Ontario

Richard Fikes
Artificial Intelligence Center
Stanford Research Institute

Michael Foster
Royal Radar Establishment

Gordon Goldstein
Office of Naval Research

Cordell Green
Artificial Intelligence Laboratory
Stanford University

Peter Hart
Artificial Intelligence Center
Stanford Research Institute

Carl Hewitt
Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Robert Kowalski
School of Artificial Intelligence
University of Edinburgh

Donald Michie
School of Artificial Intelligence
University of Edinburgh

Nils Nilsson
Artificial Intelligence Center
Stanford Research Institute

Bert Raphael
Artificial Intelligence Center
Stanford Research Institute

Jeff Rulifson
Xerox Palo Alto Research Center

Earl Sacerdoti
Artificial Intelligence Center
Stanford Research Institute

Gerald Sussman
Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Arthur Thomas
Artificial Intelligence Laboratory
Stanford University

Richard Waldinger
Artificial Intelligence Center
Stanford Research Institute

III SUMMARY OF TALKS

The following summary gives a few paragraphs outlining the substance of each talk and some of the discussion topics.

Proving Theorems about LISP Functions--Robert Boyer

This talk was based on recent work by R. Boyer and J Moore. A paper of the same title will be presented at IJCAI-73 in August at Stanford. The authors' abstract to this paper is as follows:

> We describe some simple heuristics combining evaluation and mathematical induction which we have implemented in a program that automatically proves a wide variety of theorems about recursive LISP functions. The method the program uses to generate induction formulas is described at length. The theorems proved by the program include that REVERSE is its own inverse and that a particular SORT program is correct. APPENDIX B contains a list of the theorems proved by the program.

QLISP--Earl Sacerdoti

QLISP is an extension of LISP that incorporates many features of the QA4 language[1]* including pattern matching, associative retrieval of expressions, pattern-directed function invocation, backtracking, and demons. A version is now available at SRI. Future versions will make use of the Bobrow-Wegbreit[4] control structure to implement "processes." QLISP allows easy mixing of conventional LISP and QA4-like programs and, in addition, gives the user the full power of the editing and assistance

*References are listed at the end of this report.

features of BBN-LISP (now called INTERLISP). Sacerdoti's presentation raised the issue of whether it is better to design completely a new language with all of the desired features well-integrated in a consistent fashion or to follow the QLISP approach of adding new features to an existing language, thus complicating the syntax. Most participants felt that there have been so many years of effort put into BBN-LISP, that it would be unwise to try to build a brand new language with QA4-like features.

QLISP is now being used in writing programs to prove the correctness of programs. Sacerdoti distributed a simple illustrative QLISP program that shows some of the features. This program is reproduced in Figure 1. The current version of QLISP is described in an SRI Artificial Intelligence Center Technical Note entitled "A Preliminary QLISP Manual," by René Reboh and Earl Sacerdoti, August 1973.

New A.I. Language Features for Robot Planning and Automatic Programming--Richard Waldinger

Waldinger discussed some proposed features that would be desirable additions to new A.I. languages. One is the "world-splitting" feature. Here a context is split into two contexts differing only in that there is a different assertion or goal in each. Waldinger distinguishes between an AND split and an OR split. In an AND split, the goal in each of the two contexts must both be established; in an OR split, only one of the goals need be established. Such a feature builds right into the language the capability of setting up AND/OR search trees. If accompanied by a feature that enables the programs to run in the different contexts as processes, then the supervision of search is also automatic. There are, of course, the usual problems of communication between contexts.

Waldinger also discussed the desirability of having an automatic

```
;  <SACERDOTI>QGENESIS.;17   WED 9-MAY-73 3:05PM

  (PROGN (LISPXPRIN1 (QUOTE "FILE CREATED ")
                     T)
         (LISPXPRIN1 (QUOTE " 9-MAY-73 15:05:23")
                     T)
         (LISPXTERPRI T))
(DEFINEQ

(SETUP
  [LAMBDA NIL                                          (* INITIALIZATION
                                                       ROUTINE.)
    (QASSERT (PERSON MARY)
             SEX FEMALE AGE 30 HOBBIES (CLASS TENNIS NEEDLEPOINT
                                              DANCING))
    (QASSERT (PERSON ALICE)
             SEX FEMALE AGE 72 HOBBIES (CLASS SCUBA-DIVING
                                              BIRD-WATCHING))
    (QASSERT (PERSON EVE)
             SEX FEMALE AGE 29 HOBBIES (CLASS SNAKE-CHARMING GARDENING
                                              VOLLEYBALL))
    (QASSERT (PERSON ADAM)
             SEX MALE AGE 30 NETWORTH 500000 HOBBIES
             (CLASS HUNTING FISHING GARDENING))
    (QASSERT (PERSON SARA)
             SEX FEMALE AGE 42 NETWORTH 2000000])

(MAKEHAPPY
  [LAMBDA (L)                                          (* 'L IS A LIST OF
                                                       PERSONS.)
                                                       (* TRY TO MAKE EACH
                                                       PERSON HAPPY.)
    [MAPC L (FUNCTION (LAMBDA (X)
              (PRINT (! (QATTEMPT (QGOAL (HAPPY (* X))
                                         APPLY
                                         (TUPLE HITCH RICH]
    (QUOTE FINISHED])
)
  (LISPXPRINT (QUOTE QGENESISFNS)
              T)
  (RPAQQ QGENESISFNS (SETUP MAKEHAPPY))
  (LISPXPRINT (QUOTE QGENESISVARS)
              T)
  (RPAQQ QGENESISVARS ($MARRIAGEDEMONS $COMPUTERELATIONS
                                       (P (QSETUP QGENESISVARS))
                                       (P (DEFTYPE (QUOTE MARRIED)
                                                   (QUOTE CLASS)))
                                       (FNS HITCH CHECKAGE CHECKHOBBY
                                            PARTNERSEX RICH MAKESPOUSE))
          )
  (RPAQQ $MARRIAGEDEMONS (TUPLE CHECKAGE CHECKHOBBY))
  (RPAQQ $COMPUTERELATIONS (TUPLE MAKESPOUSE))
  (QSETUP QGENESISVARS)
  (DEFTYPE (QUOTE MARRIED)
           (QUOTE CLASS))
(DEFINEQ
```

FIGURE 1

```
; <SACERDOTI>QGENESIS.;17   WED 9-MAY-73 3:25PM

(HITCH
  [QLAMBDA (HAPPY ←HUMAN)                          (* CYCLE THROUGH ALL
                                                   MEMBERS OF THE OPPOSITE
                                                   SEX.)
                                                   (* HYPOTHESIZE A
                                                   MARRIAGE AND SEE IF IT
                                                   WORKS OUT.)
                                                   (* IF IT DOES, THEN THE
                                                   HUMAN IS HAPPY.)
           (QBEXISTS (PERSON ←Y)
                     SEX
                     [=(PARTNERSEX (@ (PERSON $HUMAN]
               THEN (QASSERT (MARRIED $HUMAN $Y)
                             APPLY $MARRIAGEDEMONS)
                    (QPUT (PERSON $HUMAN)
                          MARRIEDTO $Y WRT GLOBAL APPLY
                          $COMPUTERELATIONS)
                    (@ (HAPPY $HUMAN])

(CHECKAGE
  (QLAMBDA (MARRIED ←COUPLE)                       (* MAKE SURE THE WIFE IS
                                                   NOT TOO MUCH OLDER THAN
                                                   THE HUSBAND.)
           (QPROG (←SEX
                   ←AGE
                   MALEAGE FEMALEAGE)
                  [QMAPC $COUPLE (FUNCTION (LAMBDA (SPOUSE)
                                   (QMATCHQ (TUPLE ←SEX
                                                   ←AGE)
                                            (QGET (PERSON (↑ SPOUSE))
                                                  SEX AGE))
                                   (IF (EQ $SEX (QUOTE MALE))
                                       THEN (SETQ MALEAGE $AGE)
                                     ELSE (SETQ FEMALEAGE $AGE]
                  (IF (GREATERP FEMALEAGE (PLUS MALEAGE 5))
                      THEN (QFAIL CALLER))
                  (QRETURN OK))))

(CHECKHOBBY
  (QLAMBDA (MARRIED ←X
                    ←Y)                            (* FIND AT LEAST ONE
                                                   HOBBY IN COMMON,
                                                   OTHERWISE FAIL.)
           (QATTEMPT (QMATCHQ (TUPLE (CLASS ←H
                                            ←←OTHERS)
                                     (CLASS ←H
                                            ←←OTHEROTHERS))
                              (TUPLE (QGET (PERSON $X)
                                           HOBBIES)
                                     (QGET (PERSON $Y)
                                           HOBBIES)))
             ELSE (QFAIL CALLER))))
```

```
; <SACERDOTI>QGENESIS.;17   WED 9-MAY-73 3:05PM

(PARTNERSEX
  (QLAMBDA ←X                                          (* FIND THE OPPOSITE SEX
                                                       OF THE PERSON IN
                                                       QUESTION.)
           (SELECTQ (QGET $X SEX)
                    (MALE (QUOTE FEMALE))
                    (FEMALE (QUOTE MALE))
                    (ERROR "UNKNOWN SEX "))))

(RICH
  (QLAMBDA (HAPPY ←HUMAN)                              (* TRY TO ACHIEVE A NET
                                                       WORTH GREATER THAN ONE
                                                       MILLION.)
                                                       (* IF ACHIEVABLE, THEN
                                                       THE HUMAN IS HAPPY.)
                                                       (* THIS ROUTINE NOW ONLY
                                                       MAKES A SIMPLE CHECK
                                                       AGAINST THE DATA BASE.)
           (IF (GREATERP (QGET (PERSON $HUMAN)
                               NETWORTH)
                         1000000)
               THEN (@ (HAPPY $HUMAN))
             ELSE (QFAIL))))

(MAKESPOUSE
  [QLAMBDA (PERSON ←PERSON)                            (* TEAM MEMBER OF
                                                       $COMPUTERELATIONS.)
                                                       (* ENSURES THAT THE
                                                       SPOUSE IS NOT ALREADY
                                                       MARRIED.)
                                                       (* ASSERTS THAT THE
                                                       SPOUSE IS MARRIED.)
           (QPROG ((←SPOUSE
                     (QGET (PERSON $PERSON)
                           MARRIEDTO)))
                  (IF (NOT (EQ (QGET (PERSON $SPOUSE)
                                     MARRIEDTO)
                               (QUOTE NOSUCHPROPERTY)))
                      THEN (QFAIL CALLER)
                    ELSE (QPUT (PERSON $SPOUSE)
                               MARRIEDTO $PERSON)])
)
STOP
```

```
@LISP

BBN LISP-10  12-11-72 ...


HELLO, EARL.
←SYSIN(<SACERDOTI>QLISP.SYS]          Load in QLISP
QHELLO
←LOAD(QGENESIS]                       Load files with user programs
FILE CREATED  9-MAY-73 15:05:23       Since the variable QTRACEALL is set to T,
QGENESISFNS                                all QLAMBDA functions will be QTRACEd
QGENESISVARS

GC: 28                                Garbage collection of print name storage
25, 537 FREE WORDS
(HITCH REDEFINED)
(CHECKAGE REDEFINED)
(CHECKHOBBY REDEFINED)
(PARTNERSEX REDEFINED)
(RICH REDEFINED)
(MAKESPOUSE REDEFINED)
QGENESIS.;17
←$MARRIAGEDEMONS                      Net variables are treated just like
(TUPLE CHECKAGE CHECKHOBBY)                LISP variables
←PP(HITCH]                            QLAMBDA expressions are treated just like
                                           LAMBDA expressions
(HITCH
  [QLAMBDA (HAPPY ←HUMAN)  ◆◆COMMENT◆◆     ◆◆COMMENT◆◆     ◆◆COMMENT◆◆
           (QBEXISTS (PERSON ←Y)
                     SEX
                     [=(PARTNERSEX (@ (PERSON $HUMAN]
               THEN (@ASSERT (MARRIED $HUMAN $Y)
                             APPLY $MARRIAGEDEMONS)
                    (@PUT (PERSON $HUMAN)
                          MARRIEDTO $Y WRT GLOBAL APPLY
                          $COMPUTERELATIONS)
                    (@ (HAPPY $HUMAN])
(HITCH)
←SETUP]                                    Evaluate the user's initialization function
TRUE
```

```
←MAKEHAPPY((ADAM SARA))                    Try to make Adam and Sara happy
HITCH:
QA4:ARG= (HAPPY ADAM)
  PARTNERSEX:                          ——Example of function trace
  QA4:ARG= (PERSON ADAM)
  (PARTNERSEX) = FEMALE                ——Function returns 'FEMALE
  CHECKAGE:
  QA4:ARG= (MARRIED ADAM MARY)
  (CHECKAGE) = OK
  CHECKHOBBY:                          ——Function is called, but does
  QA4:ARG= (MARRIED ADAM MARY)                not return; it caused a failure
  CHECKAGE:
  QA4:ARG= (MARRIED ADAM ALICE)
  CHECKAGE:
  QA4:ARG= (MARRIED ADAM EVE)
  (CHECKAGE) = OK
  CHECKHOBBY:
  QA4:ARG= (MARRIED ADAM EVE)

GC: 8                                      Garbage collection of list storage
5072, 10182 FREE WORDS
  (CHECKHOBBY) = (TUPLE (CLASS SNAKE-CHARMING GARDENING VOLLEYBALL)
(CLASS GARDENING HUNTING FISHING))
  MAKESPOUSE:
  QA4:ARG= (PERSON ADAM)
  (MAKESPOUSE) = ADAM
(HITCH) = (HAPPY ADAM)
(HAPPY ADAM)
HITCH:
QA4:ARG= (HAPPY SARA)
  PARTNERSEX:
  QA4:ARG= (PERSON SARA)
  (PARTNERSEX) = MALE
  CHECKAGE:
  QA4:ARG= (MARRIED ADAM SARA)
RICH:
QA4:ARG= (HAPPY SARA)
(RICH) = (HAPPY SARA)
(HAPPY SARA)
FINISHED
←
```

"protect" mechanism for maintaining the truth of goals already established during attempts to establish others.

Predicate Calculus Programming--Robert Kowalski

The author's advance abstract of this talk follows:

The interpretation of predicate calculus as a programming language is a recent development made possible by two advances in the field of automatic theorem-proving. The first is the advent of more efficient theorem-proving systems, such as SL-resolution. The second is a new appreciation for the dependence of problem-solving efficiency upon the form of the axiomatic representation of the problem. This observation, stated in terms of the programming language interpretation of predicate calculus, is just a statement of the familiar fact that different programs for solving the same problem can be equivalent in meaning but very different in computational efficiency.

The programming language interpretation of predicate calculus involves regarding an axiomatisation of a problem domain as a program for solving problems within that domain. The theorem to be proved represents the input associated with a given problem. The theorem-prover acts as an interpreter, running the program for a given input. Proving that the theorem is implied by the axioms amounts to computing a solution for the problem represented by the theorem. That theorem-provers can be used for computation has been observed before, notably by Cordell Green. What is new is our thesis that, when well-written predicate calculus programs are run by an efficient theorem-prover, then the resulting search for a solution is done in a manner similar to, or more intelligent

than, that which would be done by a more conventional programming language interpreter.

During the presentation, Kowalski emphasized the following points:

(1) There are three main paradigms for automatic problem-solving, namely the problem-reduction or AND/OR tree paradigm, the theorem-proving paradigm, and the programming paradigm.

(2) One can establish an isomorphism between the theorem-proving paradigm and the others. Specifically, if one limits oneself to propositional logic, the isomorphism is with the AND/OR tree paradigm; if one allows full predicate logic, an isomorphism with the programming paradigm can be established.

(3) The AND/OR tree paradigm is inadequate since it cannot, for example, handle dependent subproblems.

(4) In the predicate logic as programming paradigm, there is a nice split between semantics and pragmatics that tends to get mushed together in the ordinary programming paradigm. He (Kowalski), for the moment, is not too concerned with pragmatics--many others are already working on that.

(5) He is more interested in developing a model for problem-solving than he is in solving any particular problem.

(6) He feels that his predicate-calculus-as-programming paradigm makes both co-routining and syntactic parsing clearer, for example.

Kowalski presented some examples of the isomorphism between the predicate calculus and programming paradigms. These ideas are discussed fully in a forthcoming memo to be entitled "Predicate Logic as a Programming Language."

The discussion centered mainly around the question "Is A.I. semantics or pragmatics?" Some in the audience felt that the real content of A.I. is pragmatics, namely information about what to do next, how many solutions to find, and the like. Kowalski acknowledged the importance of pragmatics but was unwilling to say that it exhausted the subject matter of A.I. Since Kowalski's formalism ignores pragmatics, he is unable to give any instructions in the language about the order in which things are to be done. This lack was felt to be fatal by the pragmatics-is-all school. Nevertheless, Kowalski's formalization has an aesthetic appeal, and we should remember that yesterday's pragmatics is today's semantics is tomorrow's syntax.

Actor Formalisms--Carl Hewitt

From the people who brought us PLANNER, we now have ACTORS. Hewitt's goal is to establish a programming formalism in which programs can be written that both run and use existing technology and, in addition, are extendable. Thus he wants strong modularity.

Each module is an ACTOR. It is a piece of code that runs and is to be regarded as a black-box. Its insides cannot be tampered with. It is not easy for people to understand exactly what is going on with ACTORS. ACTORS do not call programs or return to other programs as in conventional formalisms; instead they pass messages. Hewitt showed how the ACTOR formalism can be used to do all the conventional things one wants a formalism to be able to do such as iteration, and he gave some examples of ACTOR-based programs.

There was not enough time for Hewitt to describe ACTORS sufficiently for us all to understand them, and it was agreed to hear more of this later in the program, which we did. The ACTOR formalism is discussed in more detail in a paper to be presented at IJCAI-73 entitled "A Universal Modular ACTOR Formalism for Artificial Intelligence," by Carl Hewitt, et al.

Freddy Assembles a Car--Harry Barrow

As part of his presentation, Barrow showed a film in which the Edinburgh robot system, FREDDY, assembled a toy car starting with a heap of parts. The system consists of a gripper arm, a television camera, and a platform that is movable under the arm and camera. The system is described in a paper to be presented at IJCAI-73 entitled "A Versatile Computer--Controlled Assembly System," by A. Ambler, et al.

The strategy programs for assembly are controlled by a Markof type system in which the next action taken depends on the truth or falsity of a list of predicates. Beyond this system there is currently no top-level planning in which sequences of appropriate actions can be synthesized.

In the vision system for FREDDY, objects are analyzed and represented by a relational structure. Curved objects are fitted with arcs of circles. The visual system is capable of recognizing each of the toy car parts in isolation; heaps of parts are recognized as such and invoke programs that pull promising objects out of the heaps for subsequent recognition.

The overall system works quite smoothly and was rather impressive.

The A.I. Situation

Britain--Donald Michie
U.S.A.--Bert Raphael
Canada--Ted Elcock

Michie reported on the study by Sir James Lighthill that was commissioned by the Science Research Council to appraise A.I. research. Lighthill's report, together with commentary by British A.I. researchers, has been published as a pamphlet by the Science Research Council and is entitled Artificial Intelligence: A Paper Symposium (April 1973).

Lighthill's report is a criticism generally of the attempt to make a separate science of A.I. and, specifically, of attempts to build robots.

The report can be faulted on several grounds, but it has led apparently to the cessation of SRC funding of the robot work being done under Michie's leadership at Edinburgh. Michie discussed the probable consequences of the Lighthill report on A.I. research in Britain.

Raphael described briefly the current ARPA attitude that A.I. research should be conducted according to soundly developed management plans. He also reported that ARPA felt that it could no longer support basic research in robotics.

Elcock is attempting to get an A.I. Council together in Canada and will soon send a report to SIGART describing a recent Canadian A.I. workshop.

Automatic Programming--Robert Balzer

As a first step toward automatic programming, Balzer has implemented a system called the Programmer's Interface. It is an interface between the BBN programming environment and other languages, such as EL/1. Whenever an evaluation is to be done, it is done in the user language. The system uses the ARPA network to get routines evaluated at centers having the language in which the routine is written. The system now works with PL/1, EL/1, and COBOL.

Balzer described a dozen or so features that he thought would be useful and currently implementable in a software production facility. These include: programmer's interface, perturbation detectors, interface specifiers, structured objects (like ACTORS), machine exercizes (to simulate a machine), code testers (path analysis programs), scaffolding (for testing unwritten code), and documentation aspects. Useful program verification and man-machine code generation facilities would probably not be available until somewhat later.

Balzer's approach toward automatic programming seems to be to

anchor in current technology one end of a spectrum of features and then to proceed gradually along this continuum toward more exotic facilities.

HACKER--Gerald Sussman

HACKER is a program, written in CONNIVER[3], that automatically generates debugged code to perform robot tasks in the MIT BLOCKS world. Starting with a library of documented routines (that can initially be as large or as small as you please), HACKER uses various sorts of programming knowledge--including the documentation contained in the routines--to construct a program to perform a task. In constructing the program, HACKER tries for the simple kill first using a "maybe-this-will work" approach. Tentative programs are then tested, and if bugs occur (typically they will), more detailed programming knowledge is used to find the reason for the bug and to eliminate it.

HACKER really is an attempt to simulate how a programmer (specifically Sussman) programs. It is thoroughly described in a forthcoming MIT dissertation by Sussman.

Modelling, Question-Answering and Route Finding in a Robot World--Richard Fikes

Fikes discussed some new retrieval routines he has added to QA4. These allow deduction processes to be called to attempt to prove statements not explicitly found in the QA4 data base. Such routines will be quite useful in question-answering systems.

A hierarchical modelling scheme was also described. As applied to a robot world, the hierarchy contains several levels of detail of various locations in the robot world. Fikes then described a special route-finding package for robot navigation to be used in conjunction with the hierarchical model.

Automatic Programming Research at Stanford--C. Cordell Green

In the absence of Jack Buchanan (formerly a student at Stanford), Green described the problem-solving system constructed by Buchanan and David Luckham. An abstract of the talk Buchanan was to have given follows:

> The objectives of this research project have been to develop methods and to implement a system for automatic generation of programs. The problems of writing programs for robot control, symbol manipulation and simple numerical computations have been studied and some elementary programs generated. A particular formalism, i.e., a Semantic Frame, has been developed to define the programming environment and permit the statement of a program. A Semantic Frame includes a particular representation of the world in the form of axioms and a set of fully instantiated literals, and rules for manipulating the state of the world. Rules may define atomic operators, i.e., operator rules, iterative processes, i.e., iterative rules, or situations defined as equivalent to the achievement of a set of goals, i.e., definitional rules. A Semantic Frame may be translated into a particular set of programs used in a subgoaling system which involves backtracking from the goal to be achieved to the initial state. The system is interactive, i.e., responds to some simple advice and allows incremental and "stepwise" program development. The output program or solution will transform the initial state into one which the desired goal is true. The program constructs used are procudure calls, assignments, "while" loops, and conditional statements. Some elements of the underlying logical theory for such a system will be described: the basic problem solving algorithm will be whown to be correct; and methods

for implementing the various system features will be discussed.

Green also discussed some of the approaches being taken toward automatic programming by him and by his students at Stanford. Of particular interest is a study in how to generate a program from a trace of its execution.

ABSTRIPS--Earl Sacerdoti

A problem domain may be represented as a hierarchy of abstraction spaces, in which successively finer levels of detail are introduced. The problem solver ABSTRIPS, a modification of STRIPS[4], can define an abstraction space hierarchy from the STRIPS representation of a problem domain and can use the hierarchy in solving problems.

ABSTRIPS has achieved a significant increase in problem-solving power over STRIPS. Sacerdoti illustrated this increase by an example in which the search space explored by STRIPS was much branchier than the rather direct solution to the same problem obtain by ABSTRIPS.

The ABSTRIPS program is the subject of a paper presented at IJCAI-73 entitled "Planning in a Hierarchy of Abstraction Spaces," by E. Sacerdoti.

Hierarchical Planning and Execution--Nils Nilsson

Nilsson described a program for hierarchical robot plan generation and execution. The system is further described in a note entitled "A Hierarchical Robot Planning and Execution System," by Nils J. Nilsson, Stanford Research Institute Artificial Intelligence Center Technical Note 76, April 1973. The abstract for this note follows:

> This report describes a robot control program consisting of a hierarchically organized plan generation and execution system. The program is written in QA4 and makes use of several features of that language. The usually sharp distinction between robot plan generation and execution is intentionally blurred

in this system in that planning and execution phases occur intermixed at various levels of the hierarchy. The system currently exists as a running program that clearly illustrates the concepts involved; major additions and refinements would be necessary if the system were to be used to control an actual robot device.

Harry Barrow mentioned that some similar work was being performed at Edinburgh by a student, Philip Hayes. Hayes' work concerns the hierarchy of planning and execution performed by a travel agent in arranging trips.

The discussion then focused on a sample planning and execution task used in a Stanford University seminar in A.I. languages. Some controversy ensued over whether these kinds of problems could be studied adequately with robot simulations or, instead, required robot (particularly vision) hardware.

"Small Talk"--J. F. Rulifson

"Small Talk" is a programming language being developed by Alan Kay and Jeff Rulifson at Xerox Palo Alto Research Center for use in teaching children about computing. It shares several ideas with the ACTOR formalism, and Rulifson spent most of his time talking about the similarities and differences between ACTORS and "Small Talk."

Robotic Equipment--Nils Nilsson

Nilsson briefly reviewed the preliminary conclusions of an SRI study on robotic equipment. These are contained in an interim report being written for the National Science Foundation entitled "Study of Equipment Needs for Artificial Intelligence Research," by N. J. Nilsson, et al. The abstract of this report follows:

This report describes interim results of a project to specify

special equipment for research in Artificial Intelligence. After surveying several potential users it was decided that there was a need for standardized equipment for robot research. Such equipment includes a vision subsystem, an arm subsystem, and a moblie cart subsystem. Some users desire a robot consisting of all three sybsystems, while others need lesser combinations. Taking into account these user needs, and equipment cost and availability considerations, we recommend a list of modular components that in total comprises a complete mobile robot system. Alternatively, subsets of the component list can be used to suit particular user needs. The recommended systems include a small minicomputer to be used for arm trajectory, vehicle navigation, and interfacing with the user's main computer. Use of the minicomputer provides needed research flexibility in that control algorithms can be easily revised and additional user sensor and effector equipment can be easily added.

The complete vehicle/arm/vision robot system will be controlled from the user's main computer over a radio link to to the on-board minicomputer. The robot will be able to operate in an office or laboratory environment, and will be powered by rechargeable storage batteries. Television pictures from the vehicle will be sent back to the main computer over a video-bandwidth radio link.

Cost estimates for both the prototype equipment and future copies are given.

IV CONCLUSIONS

At the end of the workshop, it was the general consensus that such informal get-togethers provide an extremely valuable means for learning about and discussing each other's work. At this workshop, many of us were surprised to see that so many problem-solving issues are now being discussed as issues of A.I. language design. Since this trend of incorporating problem-solving strategies directly into A.I. languages will probably continue, we concluded that any future workshop on automatic problem-solving ought to include, as this one did, specifically recent work in A.I. languages.

Most of the participants agreed that much the same group should meet again for another workshop next year. Donald Michie suggested that perhaps the next one could again be in Scotland, possibly immediately preceding or following IFIP-74.

REFERENCES

1. J. F. Rulifson, Jan Derksen, and R. J. Waldinger, "QA4: A Procedural Calculus for Intuitive Reasoning," Technical Note 73, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California (November 1972).

2. D. G. Bobrow and B. Wegbreit, "A Model and Stack Implementation of Multiple Environments," Report No. 2334, Bolt, Beranek, and Newman, Incorporated, Cambridge, Massachusetts (1972).

3. G. J. Sussman and D. V. McDermott, "From PLANNER to CONNIVER--a Genetic Approach," AFIPS Conference Proceedings, Vol. 41, Part II, pp. 1171-1179, 1972 Fall Joint Computer Conference (AFIPS Press, New Jersey 1972).

4. R. E. Fikes and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," Artificial Intelligence, Vol. 2, pp. 289-308 (1971).

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)*	2a. REPORT SECURITY CLASSIFICATION
Stanford Research Institute 333 Ravenswood Menlo Park, California 94025	Unclassified
	2b. GROUP N/A

3. REPORT TITLE

PAJARO DUNES WORKSHOP ON AUTOMATIC PROBLEM SOLVING

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

Final Report Covering the Period 29 January 1973 to 30 June 1973

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

Nils J. Nilsson

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
July 1973	26	4

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-73-C-0245 b. PROJECT NO.	Final Report SRI Project 2527
c. d.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Office of Naval Research 800 North Quincy Arlington, Virginia 22217

13. ABSTRACT

This report describes the talks and discussions occurring at a Workshop on Automatic Problem Solving held at Pajaro Dunes, California on May 14-16, 1973.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6801

UNCLASSIFIED

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Problem solving						
Robots						
Artificial Intelligence Languages						