EMERALD 2 AN INTEGRATED SYSTEM OF MACHINE LEARNING AND DISCOVERY PROGRAMS FOR EDUCATION AND RESEARCH USER'S GUIDE

by

K. Kaufman
A. Schultz
R. S. Michalski

Reports of the Machine Learning and Inference Laboratory, MLI 93-8, School of Information Technology and Engineering, George Mason University, Fairfax, VA, September 1993.

EMERALD 2:

An Integrated System of Machine Learning and Discovery Programs for Education and Research User's Guide

K.A. Kaufman, R.S. Michalski and A. Schultz

MLI 93-8

EMERALD 2: An Integrated System of Machine Learning and Discovery Programs for Education and Research

USER'S GUIDE

Kenneth A. Kaufman, Ryszard S. Michalski and Alan C. Schultz

Center for Artificial Intelligence
George Mason University
4400 University Drive
Fairfax, VA 22030

Abstract

EMERALD 2 is a large-scale system integrating several advanced programs exhibiting different forms of learning or discovery. The system is intended to support teaching and research in the area of machine learning. It enables a user to experiment with the individual programs, run them on various problems, and test the performance of the programs. The problems are defined by a user from a set of predefined visual objects, displayed through color graphics facilities. The current version of the system incorporates the following programs, each displaying the capacity for some simple form of learning or discovery:

AQ - learns general rules from examples of correct or incorrect decisions made by experts.

INDUCE - learns structural descriptions of groups of objects and determines important distinctions between the groups.

CLUSTER - creates meaningful categories and classifications of given objects, and formulates descriptions of created categories.

SPARC - predicts a possible continuation of a sequence of objects or events by discovering rules characterizing the sequence observed so far.

ABACUS - conducts experiments, collects data, formulates mathematical expressions characterizing the data, and discovers scientific laws.

Individual programs, presented as robots, communicate their results by natural language sentences displayed on the screen, and by voice. EMERALD 2 is an extension of ILLIAN, a smaller system developed for the exhibition "Robots and Beyond: The Age of Intelligent Machines". The exhibition was organized by a consortium of major US Museums of Science. The system was initially implemented for a DEC VaxStation, while the current version of the system was developed for use on a Sun workstation.

Acknowledgements

The EMERALD system integrates several machine learning and discovery programs that have been developed on the basis of research conducted by the research group of R.S. Michalski over the span of about twenty years, first at the University of Illinois at Urbana-Champaign, and then at the Center for Artificial Intelligence at George Mason University. The Center's research is supported in part by the National Science Foundation under the grant No. IRI-9020266, in part by the Advanced Research Projects Agency under the grant No. N00014-91-J-1854, administered by the Office of Naval Research, and under the grant No. F49620-92-J-0549, administered by the Air Force Office of Scientific Research, and in part by the Office of Naval Research under the grant No. N00014-91-J-1351.

The support for the research that led to the development and continuing improvement of these programs was provided by a number of grants received over the years, specifically, by the National Science Foundation under grants No. MCS-74-03514, MCS-76-22940, MCS-79-06614., MCS-82-05166, MCS-83-06614, and NSF DCR 84-06801; by the Office of Naval Research under grants No. N00014-81-K-0186, N00014-82-K-0186, N00014-88-K-0186, N00014-88-K-0226, N00014-88-K-0397 and N00014-91-J-1351; and by the Defense Advance Research Projects Agency under grants administered by the Office of Naval Research No. N00014-85-K0878, N00014-87-K-0874 and N00014-91-J-1854. We particularly acknowledge the Naval Research Laboratory for supporting the effort of A. Schultz in the adaptation of EMERALD to the SUN workstation.

The support for the development of ILLIAN, the initial smaller version of EMERALD that was presented at the exhibit "Robots and Beyond: the Age of Intelligent Machines," was provided by the Digital Equipment Corporation, the Boston Museum of Science, and the University of Illinois at Urbana-Champaign. The development team of the initial version was led by R. S. Michalski, with assistance from Professors R. E. Stepp and D. Medin, and included A. Buriks, T. Channic, K. Chen, A. Gray, G. Greene, C. Kadie, K. Kaufman, H. Ko and P. Ong.

The EMERALD system was developed at the George Mason University Center for Artificial Intelligence under the direction of R. S. Michalski, in collaboration with K. DeJong, K. Kaufman, A. Schultz, P. Stefanski and J. Zhang. We also gratefully acknowledge the support of Professor D. Rine, Chairman of the GMU Department of Computer Science, and the help in testing and experimenting with the system from the members of the AI Center, including E. Bloedorn, J. Bala, K. Dontas, R. Hamakawa, J. Wnek and M. Wollowski.

Table of Contents

1 INTRODUCTION

This report describes the basic architecture and use of an integrated system of machine learning and discovery programs, called EMERALD, intended to demonstrate machine learning capabilities, and to serve as a tool for education and research in machine learning. To serve these needs, two systems have been developed:

ILLIAN - an initial, short version used specifically for demonstrating machine learning capabilities. This version was developed for the exhibition "Robots and Beyond: The Age of Intelligent Machines", organized by a consortium of major U.S. Museums of Science (Boston, Philadelphia, Charlotte, Fort Worth, Los Angeles, St Paul, Chicago and Columbus), and was presented at those museums during the years 1987-1989. This earlier version is now on permanent exhibit in Boston.

EMERALD (**E**xperimental **M**achine **E**xample-based **R**easoning and **L**earning **D**isciple) - an extended version for use as both an educational tool in machine learning and related areas, and as a laboratory for experimentation and research. Two versions of the system are currently available: EMERALD/M, which runs on the DEC VaxStation, and EMERALD/S, which runs on the Sun workstation.

The system is based on many years of research done by Michalski's research group in the area of machine learning. As this area has recently become very active, and many researchers have started to work in machine learning, we felt it would be of interest to the scientific community to integrate different programs into one system, and make it available for use in education and research. The integrated system makes it easy for a user to interact with and run individual programs, test them, and acquire experience in understanding their functions and their capabilities.

A complete, seamless integration of these programs is a very difficult task, requiring significant modification of the input and output modules of these programs and the development of a complex control mechanism. As the first step toward such a goal the programs were integrated at a user level, i.e., the system allows a user an easy access to each program through a menu and facilitates an application of each to problems defined by the user, employing various predefined objects.

The capabilities of the EMERALD include the ability to learn general concepts or decision rules from examples, create meaningful classifications of observations, predict sequences of objects, and discover unknown mathematical laws. A user may be surprised by some capabilities of the programs. Sometimes a user may do better than the machine, but sometimes it may be the machine that does better.

The examples used in the demonstration deal with very simple objects -- pictures of imaginary robots or trains, geometrical figures, cards, etc. -- so that anyone can easily understand them. These learning programs, however, have already been applied to, and have a potential to be useful in many areas, such as medicine, agriculture, biology, chemistry, financial decision making, computer vision, database analysis, and, of course, intelligent robotics.

The current system, EMERALD 2, integrates the following five programs, presented as robots, each displaying a capability for some simple form of learning or discovery:

AQ -- learns general decision rules from examples of correct or incorrect decisions made by experts.

INDUCE -- learns descriptions of groups of objects and determines important distinctions between the groups.

CLUSTER -- creates meaningful categories and classifications of given objects.

SPARC -- predicts possible future objects in a sequence by discovering rules characterizing the sequence observed so far.

ABACUS -- formulates scientific laws and discovers mathematical patterns in data.

This report provides a guide to installation and basic use of the EMERALD system.

The next chapter, Chapter 2, describes how to run EMERALD. Chapter 3 gives an overview of the use of the individual programs, and the different options for running experiments with them.

2 GETTING STARTED WITH EMERALD

The EMERALD system runs in a Common Lisp environment on the Sun Workstation. It requires OpenWindows software and a monitor with full color graphics, and a DECTALK voice synthesis module is optional, but highly recommended. The latter device should be hooked into a serial port corresponding to /dev/tty00 on the Vax, or /dev/ttya on the Sun. In addition, EMERALD requires a Sun Pascal library in order that the two Pascal-based learning modules (AQ and SPARC) may run successfully. See **Appendix A** for a full description of the installation procedure.

In order to run EMERALD on a Sun machine, it is necessary to first create a disk image of a Common Lisp environment under OpenWindows (this environment is referred to as CLX.) Assuming that this has been done, the user must first invoke the OpenWindows system from the EMERALD home directory using the *openwin -noauth* command, bring up the CLX disk image (the example given in Appendix A will be in a file called **clx-lisp**), and then tell the Lisp system to load the file *emerald.lisp*. This file initializes the EMERALD environment, and loads all the files necessary for the system to run. This loading takes under two minutes. Early in the loading period the user will be asked whether the host is a Sun 3 or a Sun 4, whether to enable the talk facility, and whether or not to use the local system for display. Normally, the latter question should be responded to affirmatively, but this option allows for the case of networked Suns in which EMERALD can run on one processor, while the display and user interface appear on another machine. Finally, the user will be asked about debugging mode. In mode 0, a general message screen appears if EMERALD has encountered a fatal error, after which the system will restart. In mode 2, the program will exit, and the specific error message will appear in the Lisp window. Note: It may be necessary to bring that window to the front. After this is answered, the display screen will go dark.

When the loading has finished, the initial screen will automatically come up. Chapter 3 describes how to proceed through EMERALD.

After the user has finished running EMERALD, the windows that had been on the screen will remain in the background; the user should push the EMERALD window to the back. The window running the Lisp process will display the function call -- (RESTART) -- required to restart the system, and will then return to the command level. The user can then run another EMERALD session, interact with the Lisp interpreter, or quit the Lisp environment by the

standard (QUIT) function.

3 OVERVIEW AND USER'S GUIDE

This section discusses the nature of an EMERALD session from the user's point of view. It is designed to stand independently as a user's guide to EMERALD, and also to serve as supplementary information to assist programmers and maintainers in their understanding of EMERALD.

The original desire of the EMERALD project was to create an exhibit displaying a set of pages (slides) through which the user could progress in much the manner one pages through a book. To this end, the system was structured in much the way one organizes a text. The system presently consists of five different programs, called learning robots:

AQ INDUCE CLUSTER SPARC and ABACUS

Each consists of at least the following three subprograms:

Robot Challenges You (Introduces user to abilities of Robot)
You Challenge Robot (Allows user to experiment with Robot)
Find out How Robot Works (Explains briefly the theory behind its operation)

Different robots may have additional submodules. As the user progresses through the exhibit, two types of pages may be encountered. These types are "menu" pages and "work" pages. A menu page consists solely of information and choices, while a work page allows the user to interact with the exhibit in a manner necessary for a particular application. To the best possible extent, the final code has remained faithful to this initial goal. The remainder of this section describes the contents of the hypothetical book which the exhibit models.

The program runs in a continual loop; as long as a certain escape sequence is not applied, the system will return to its initial screen whenever either:

- a user tells it to restart
- it has remained untouched for a certain amount of time
  (thereby causing the time-out procedure to be invoked)

This screen displays the title of the demonstration with full-color graphics illustrating some of

the icons encountered throughout execution. It also indicates, both by print and by voice, how the user may proceed with the demonstration. It is in this state that EMERALD waits for a user. This first menu page represents the cover of EMERALD exhibit.

The next page of the exhibit, labelled MAIN MENU, displays the contents of the exhibit. The page contains an image of EMERALD and each of the subordinate robots which the user can visit. On this page, the user first encounters the standard set of choice squares located at the bottom of the screen. These squares are labeled "QUIT THE EXHIBIT", "RESTART THE EXHIBIT", "HELP", "BACK ONE SCREEN", and "NEXT SCREEN" on this particular page. Similar boxes will be present on all other screens, with "VISIT ANOTHER ROBOT" added and"<robot> MAIN MENU" replacing the restart square. The purpose of these squares is to provide standard methods by which one may progress through the exhibit. From the main menu, the user may choose to visit any of the five robots by selecting one of these robots, or choosing from the bottom of the screen. If NEXT SCREEN is chosen, the user will begin with the first chapter (AQ) and eventually progress from left to right through the robots shown on this page.

3.1 AQ ROBOT

Upon entering the chapter describing AQ, one first encounters a menu allowing the selection of one of the three standard subsections:

AQ Challenges You
You Challenge AQ (See Note) or
Find Out How AQ Works

NOTE: At present the You Challenge AQ portion is divided into a simple and advanced portion at the top level. As the system is expanded, this division will eventually be made after the selection of the You Challenge AQ option.

As is typical of each chapter, the name of the present chapter and an image of its representative robot are found at the top of the page. Again, the choice of NEXT SCREEN chooses the first of these options.

3.1.1 Option: *AQ Challenges You*

This section presently contains seven menu pages - three pairs of problem/solution pages, and a summary page. Each of the three problem/solution pairs is comprised of a problem which challenges the user, followed by a page giving the solution to the problem. All problems are of a similar form, asking the user to find a rule which properly distinguishes between friendly and unfriendly robots. The first page is slightly more detailed than the others, since it describes the hypothetical domain in which the user will be asked to solve problems (i.e. a world of friendly and unfriendly robots.) Solutions are shown in green rectangles (rule blocks) and are presented on the page following the presentation of the problem. The problems, which grow progressively more difficult, are provided to familiarize the user with the types of problems AQ will solve later in the exhibit. The summary page gives a brief description of the significance of the preceding problems and explains how such reasoning can be extended to other domains. After this list of possible applications, the user is asked to challenge AQ.

3.1.2 Option: *You Challenge AQ with a Simple/Advanced Problem*

SIMPLE PROBLEM

This section consists of a menu page and a work page. The first page introduces the problem to the user. This is especially useful for users who have not visited the previous section. The second page allows the user to interact with AQ. It prompts the user to form two sets of robots for which AQ will find a distinguishing rule. The two sets are introduced intuitively as members and non-members of a club which the user must define. The user may then choose the yellow area labelled DISCOVER RULE, at which time AQ finds a distinguishing rule. At this point, the form of the work page is altered to allow presentation of the rule. In addition to displaying the new rule and stating it using the voice of AQ, the new page allows the user to either:

SEE AN ALTERNATIVE RULE	(i.e. another rule to distinguish members from non-members)
ADD OR DELETE ROBOTS	(i.e. modify the present problem)
MAKE A NEW PROBLEM	(i.e. restart work page)

The user can leave the work page by selecting NEXT SCREEN.

COMPLEX PROBLEM

This section, as does the previous one, consists of a menu page and a work page. The menu page allows the user to specify the number of classes of robots, a value from 1 to 4. The following work page is analogous to that of the SIMPLE section except that there is room for up to 4 classes of robots. The page on which the rule is displayed, however, may contain two options not present in the SIMPLE PROBLEM section. These are:

DELETE GROUPS (i.e. decrease the number of classes)
CREATE GROUPS (i.e. increase the number of classes)

The Delete Group option will appear whenever there is more than one group currently defined, and the Create Group option will appear whenever there are fewer than four groups defined.

3.1.3 Option: *Find Out How AQ Works*

This section consists of a single menu page. The page, as is typical of other "find out how it works" pages, describes the manner in which AQ works and its significance in real-world applications.

3.2 INDUCE ROBOT

The INDUCE chapter begins with a menu page displaying three options:

See examples of what INDUCE can do
You challenge INDUCE to discover a concept
Find out how INDUCE works

3.2.1 Option: *INDUCE Challenges You*

The first page in this section introduces three options:

EX 1: What is an arch?
An example of incremental and non-incremental learning.

EX 2: How to distinguish groups of objects?
Distinguishes between multiple groups.

EX 3: INDUCE discovers patterns in trains
Introduces the user to the domain in which INDUCE may be challenged.

Of these, only Example 1 does not actively challenge the user, instead presenting the learning process in a step-by-step tutorial manner.

EXAMPLE 1: What is an Arch?

This example consists of four menu pages. The first two pages describe incremental learning, the next page describes non-incremental learning, and a final page provides a summary describing the importance of INDUCE. Incremental and non-incremental learning are described by an example which explores how the concept of an arch might be learned using each method. Each page presents figures of arches and describes why particular rules either do or do not capture the concept of an arch.

EXAMPLE 2: How to Distinguish Groups of Objects?

This example consists of a single work page challenging the user. The user is asked to devise a rule to distinguish between three sets of colored two-dimensional objects. Three possible rules are presented, and the user is asked to select one of these three possibilities. Upon selection of the correct rule, or upon the user's selection of one of the standard options, the user leaves the page.

EXAMPLE 3: INDUCE Discovers Patterns in Trains

This challenge consists of four menu pages grouped as two sets of problem/solution menu pages. Each problem asks the user to devise a rule to distinguish between two sets of trains. The rule discovered by INDUCE is then presented on the solution page. The first problem is an example of a simple rule, which may not be obvious to the user. The second problem is analogous to the historic train example presented by Winston.

3.2.2 Option: *You Challenge INDUCE*

This portion of the INDUCE chapter consists of a single work page. The page begins with a brief introduction describing how to use the screen, followed by a white region representing a set of 48 trains. Because only 8 trains can fit into this region, yellow bars with arrows are

placed on the right and left of these trains to allow the user to scroll through the different pages of train images. The white region on the bottom of the screen consists of two sets of four slots into which trains from the above section can be placed. Trains placed in the left four slots represent SAFE trains while those in the right represent UNSAFE trains (this is made obvious by labels printed next to the slots). Upon having placed trains in each of these two classes, the user can challenge INDUCE to discover patterns distinguishing the two sets of trains by selecting DISCOVER PATTERN. Upon making this choice, INDUCE devises a rule to describe SAFE trains. After presenting the rule devised in the area previously occupied by the set of trains, INDUCE gives the user three choices:

See an Alternative Rule (i.e. another rule distinguishing the two classes)
Add or Delete Trains (i.e. allow the user to modify the problem)
Create a New Problem (i.e. restart the work page)

3.2.3 Option: *Find Out How Induce Works*

This section consists of a single menu page which describes the manner in which INDUCE operates and the significance of INDUCE as a learning program.

3.3 CLUSTER ROBOT

This section begins with a menu page displaying the following options:

CLUSTER challenges you with a problem
You challenge CLUSTER with a problem
Find out how CLUSTER works

3.3.1 Option: *CLUSTER Challenges You With A Problem*

This section consists of two pages that present an example of one type of problem that CLUSTER can solve. The first page introduces a simple problem in which the user is asked to place a set of eight geometric figures into groups. The second page is a work page. This page allows the user to select between three possible groupings of the eight objects. By placing the

cursor controlled by the mouse over either a particular clustering or one of the yellow bars respectively labelled First Grouping, Second Grouping or Third Grouping, the user is able to select the grouping found to be most appealing. After making a choice, the user is informed whether or not the chosen clustering agrees with that which CLUSTER would (hypothetically) select. If the user fails to select the correct clustering, the system allows the user to try until the correct grouping is selected. At this point the selections available on the upper section of the page are disabled and only the menu selections at the bottom of the screen may be chosen.

3.3.2 Option: *You Challenge CLUSTER With A Problem*

This section consists of a single work page. The page contains an upper portion with 12 trains, from which the user is asked to select a subset consisting of two to eight trains. These can be moved to the lower portion of the screen, the EXAMPLE SET, which contains 8 positions for trains. Trains are moved by first placing the selection cursor over a train with the mouse and pressing the select button, then moving the cursor to another white rectangle and again pressing the select button. After more than one train has been placed in the EXAMPLE SET, the user may select the yellow region labelled FORM GROUPS. Upon this choice, CLUSTER begins attempting to divide the trains into groups based upon properties such as the types of cars in the train and the loads they contain. Upon finding a classification, the groups formed are displayed at the the top of the screen. Each group is described by a rule, shown in a green region (rule block), which distinguishes it from the other groups. The bottom of the screen contains two selections: SEE ALTERNATIVE GROUPING and CHANGE YOUR EXAMPLE SET. Choosing the first selection causes CLUSTER to find a new set of rules partitioning the set of trains into groups. These will then be displayed in the same manner as the present groups. If the second option, CHANGE YOUR EXAMPLE SET is chosen, then the screen is returned to the state it was in before FORM GROUPS was originally chosen.

3.3.3 Option: *Find Out How Cluster Works*

This section is comprised of a single menu page, which provides a brief description of how CLUSTER works followed by a short description of the scope of problems to which CLUSTER has actually been applied.

3.4 SPARC ROBOT

This chapter begins with a menu page displaying four possible options

SPARC Challenges You: Simple Sequence
SPARC Challenges You: More Complex Sequence
You Challenge SPARC
Find Out How SPARC Works

Since SPARC is the most complex of the EMERALD learning systems, the problems it can solve may overwhelm an unsuspecting user. In order that the user may have a smooth introduction to sequence prediction, the extra option (simple sequence) has been added to the list of available selections.

3.4.1 Option: *SPARC Challenges You*

This section is made up of four distinct work pages. Each presents a problem which asks the user to complete a particular sequence (i.e. choose the next element). The domains become progressively more difficult.

The simple sequence consists of a single work page. The top of the page displays a sequence of colored geometric figures (triangles squares and circles) which the user is asked to continue. Currently, the example is always the same: Red Triangle, Blue Square, Green Circle, repeating. Possible continuations are shown in four square white regions below the sequence, and the user is asked to select the best one. Upon the first selection of an incorrect continuation, the user is told that the choice is incorrect and allowed to make another attempt. Upon selection of a either a second incorrect continuation, the correct solution or NEXT SCREEN, the work page displays the correct solution accompanied by an explanation of why that particular solution is correct. The user is then allowed to continue by selecting any one of four select bars:

SPARC Challenges You: More Complex Sequence [geometric figures]
SPARC Challenges You: Mined Channel (advanced)
SPARC Challenges You: Card Game (advanced)
You Challenge SPARC

or one of the standard options at the bottom of the menu.

Had the user selected MORE COMPLEX SEQUENCE from SPARC's main menu, another menu would appear, allowing the selection between geometric figures, mined channels, or playing cards. Both the geometric figure and playing card options consist of an arbitrarily long sequence of work pages (it can also be viewed as a single work page with a large collection of problems, of which the user may choose as many as are desired.) As in the simple sequence example, the page asks the user to continue a sequence of geometric figures or playing cards. However, in this case, an upper region contains two rows -- a Main Line (containing the sequence) and a Side Line (containing incorrect examples). If an incorrect continuation is selected, the figure chosen is displayed in the Side Line below the point in the sequence at which that error was made, and EMERALD adds the correct element to the Main Line. If the correct continuation is chosen, the series is extended by one object and the user is asked to try again. After the user has guessed (whether correctly or incorrectly) three elements to be added to the sequence, or if "NEXT SCREEN" is chosen after one or two guesses, an explanation of the sequence is displayed in a rule block (green region) and the user is allowed to decide whether to try another such problem, be challenged by SPARC in one of the other environments, to challenge SPARC with a problem, or to go elsewhere via the standard options on the bottom of the screen.

The Mined Channel example contains one description page and up to three work pages. The first page describes a problem in which a ship must break a code of beacons on the shores of a channel in order to proceed through safely. The code determines which paths are safe; other paths contain mines which must be avoided. Although not stated clearly, the beacons have three features (shape, color, and location along the path). The user is asked to determine from a part of the safe path shown on a map (presumably an enemy ship was spotted negotiating a portion of the safe route) the next direction in which the ship should sail. The choice is made by clicking the mouse on one of the arrows moving away from the ship. Depending on the selection, the ship will either pass safely through or hit a mine and sink. Either way, SPARC will display the rule it had in mind. The rules on the three Mined Channel pages remain the same during each execution of the program. They are similar in nature to those generated for the geometric figures and card game, but they are much more difficult to discover due to the volume of available information.

3.4.2 Option: *You Challenge SPARC*

This section is comprised of a single menu page and one work page for the user to choose from, involving a sequence of playing cards. This is analogous to the Card Game challenge described in section 3.4.1. However, the user, as opposed to SPARC, now creates the sequence.

The first time the user enters the work page after entering the SPARC main menu, the work page shows a partial sequence in order to give the user an idea of how to continue. Thus after this first example, a user would typically begin by thinking of a sequence of cards. These can then be created using the "COMPOSE A CARD" option found at the bottom of the page, and placed on "THE TABLE" shown at the top of the screen. The area under the title "CARD-IN-HAND" represents the card presently in the users hand. (NOTE: The user can either create the CARD-IN-HAND by using the COMPOSE A CARD section, or by picking a card from the table using the mouse and the SELECT button.) The user creates the sequence which SPARC is to continue in the top row. Negative examples are placed in a side line below the cards which they are not allowed to follow. For example, if the fourth card in the sequence (i.e. top row) may not be the three of hearts, this could be placed underneath the card in the third row that it could not legally follow.

After configuring the table in this manner, the user can choose one of the following options:

FIND RULE & PREDICT NEXT CARD
FIND AN ALTERNATIVE RULE
CLEAR TABLE

If the first option is chosen, SPARC will attempt to continue the sequence. The card SPARC proposes is shown at the end of the sequence on a background of a different color (i.e. not white). This is followed by the display of the rule (NOTE: If the rule is too long to fit on the screen, the user is given the option of either seeing the rule on a separate screen (the YES option) or ignoring it (the NO option.)) If the second option, FIND AN ALTERNATIVE RULE, is chosen, a new rule is found explaining the present configuration. Finally, CLEAR TABLE removes all cards from "the table".

There are many ways to use the negative examples effectively when SPARC guesses incorrectly. One obvious method is to place the incorrect choice in a row below its present position and optionally extend the sequence by one card. Alternatively, it can be placed in a

position below an earlier card in the sequence.

3.4.3 Option: *Find Out How SPARC Works*

This section consists of five menu pages. The first page describes the problem with which SPARC is faced. The next three describe the three rule models used by SPARC. The fifth page describes problem domains in which SPARC may be applied.

3.5 ABACUS ROBOT

The layout of ABACUS differs from that of the others primarily in that the option FIND OUT HOW ABACUS WORKS has been placed before the others, and broken into two sections. This is justified by the fact that these two sections, ABACUS DISCOVERS OHM'S LAW and ABACUS DISCOVERS STOKE'S LAW give more of a description of what ABACUS does than how it performs this feat. The remaining two sections, ABACUS CHALLENGES YOU and CHALLENGE ABACUS WITH YOUR PROBLEM, again follow the standard format.

3.5.1 Option: *Find Out About ABACUS*

The section, ABACUS DISCOVERS OHM'S LAW, consists of one work page and one menu page. The work page contains a table of data and three yellow regions containing the following candidate rules which may describe the data:

$3 \times R = V$, $2 \times I = R$, or $V = I \times R$.

The user is asked to select that rule which best describes the data. If the correct choice (i.e. the third) is not selected, a reason is given why it is inappropriate. After two attempts at this problem, the user must progress to the next screen regardless of the choices made. The menu page which follows gives a description of the kinds of areas in which ABACUS can be applied and the types of equations it can discover.

The next section, ABACUS DISCOVERS STOKE'S LAW, consists of a single menu page. The upper left corner of the screen contains an animated example of balls falling in cylinders. The screen explains that ABACUS can find a set of rules to describe a situation in which a single rule may not be appropriate.

3.5.2 Option: *ABACUS Challenges You*

This section consists of two work pages and two menu pages. The user is first presented with a work page. On the page, one attempts to angle a cannon so as to land a projectile in a box. The cannon fires a cannonball against a wall; the ball rebounds in the direction of the box. Upon a success, a menu page appears. On this page, the material of the wall and the speed at which the ball are fired are changed. At this point the user may either choose to experiment in the new environment by selecting the yellow bar TRY AGAIN or continue elsewhere by selecting one of the standard options. If the user continues to experiment (i.e. selects TRY AGAIN) a new work page appears, but with a different wall material and firing velocity. Again, success leads to the appearance of a menu page. However, this time the menu page describes the form of the rules ABACUS would uncover for the examples just solved by the user. At this point the user may continue by selecting a standard menu option.

3.5.3 Option: *Challenge ABACUS With Your Problem*

This section consists of a two work pages and a menu page. The work pages allow the user to set three parameters which define the environment. The user can position the box in which the cannon ball must land, and set the material of the wall (steel, brick, wood or plastic in order of descending elasticity) and the explosive power of the cannon (low, medium, high, or very high). Upon selection of the rectangle marked CHALLENGE ABACUS, ABACUS begins to experiment in a trial-and-error manner analogous to that in which the user had to in the previous section. After five shots at predetermined angles, the system has enough information to generate an equation as a function solely of cannon angle, and declares that a rule governing the flight of the ball has been found. If it is not possible to land the ball in the box, this is stated, and a new work page is displayed which allows the user to reposition the box and continue the simulation with the new position. The legal set of possibilities is indicated by a yellow region. The cannon is then fired several times; the ball will land in the box each time. The user is now given two choices:

See The Equation Found By ABACUS
Challenge ABACUS With Another Environment

If the the user chooses to see the equation, this is displayed on a menu page which allows the user to either challenge ABACUS with another environment or to choose a standard menu option. The second option simply allows the user to pose another problem.

APPENDIX A: INSTALLATION ON THE SUN WORKSTATION

Hardware requirements: A Sun workstation - Sun 3 or higher, with a color monitor. A 1/4 inch tape drive capable of reading high-density tapes A DecTalk voice synthesis device is optional, but highly recommended.

Software requirements: OpenWindows, version 2.0 or higher, Sun (Lucid) Common Lisp, version 4, and libraries from Sun Pascal, version 2.

The first step in installing the EMERALD system is to set up its home directory. Once it is created and moved to, copy the contents of the EMERALD tape into the directory using the tar **xvbf 1024** command (it will create its own subdirectories), and use the **chown** command to set up the ownerships of the EMERALD files. One file must be edited to take into account the location of this directory. To do this, go into the file *emerald.lisp*, and find the line that reads:

(setf *exhibit-path* "<some path>/")

Change what appears within the quotation marks to the complete path of the EMERALD home directory. Be sure to leave the trailing "/".

In addition, the Common Lisp/X-Windows interface must be created, the Pascal library must be accessible, and the system must be able to find non-standard fonts used by EMERALD. In order to achieve the former, the following steps should be performed:

Go to the CLX subdirectory of the OpenWindows directory. Enter the following sequence of commands:

lisp	*Enters Lisp environment*
(load "defsystem.lisp")	
(compile-clx)	*This takes a few minutes*
(quit)	*Exit so that space may be cleaned up.*
lisp	*Reenters Lisp environment.*
(load "defsystem.lisp")	
(load-system)	*Loads the compiled files.*
(disksave "clx-lisp")	*Saves the X-Lisp interface environment.*
(quit)	*Returns to system command level.*

Some of the above commands may require superuser privileges. In order to access all necessary fonts and the Pascal library, the following lines should be added to your .login file:

```
setenv LD_LIBRARY_PATH $OPENWINHOME/lib:/usr/local/lang/pascal
setenv FONTPATH :$OPENWINHOME/lib/fonts:<EmeraldHomeDirectory>/oldfonts:
```

where <EmeraldHomeDirectory> is the top level directory for the system.

To make the Pascal library accessible, either put it in /usr/local/lang/pascal or create a symbolic link from that directory to the library file. This file will have a name beginning with "libpc".

The system will be ready to run, as described in Chapter 2, once the path in *emerald.lisp* is specified.

REFERENCES

General:

Dietterich, T. G. and Michalski, R. S., "A Comparative Review of Selected Methods for Learning from Examples," Chapter 3 in Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Tioga Publishing Company, Palo Alto, CA, 1983, pp. 41-82.

Kodratoff, Y. and Michalski, R. S. (Eds.), *Machine Learning: An Artificial Intelligence Approach, Vol III*, Morgan Kaufmann Publishers, San Mateo, CA, 1990.

Lenat, D., "AM An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search," Computer Science Department, Rept. STAN-CS-76-750, Stanford University, Stanford, CA, 1976.

Michalski, R. S., "Pattern Recognition as Rule-Guided Inductive Inference," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, No. 4, pp. 349-361, July 1980.

Michalski, R. S. (Ed.), Proceedings of the International Machine Learning Workshop, University of Illinois Allerton House, Urbana, IL, June 22-24, 1983.

Michalski, R. S., "Concept Learning," *Encyclopedia of Artificial Intelligence*, S. Shapiro ed. John Wiley and Sons Publishers, New York, NY 1987, pp. 185-194.

Michalski, R. S. and Chilausky, R. L. , "Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis," *International Journal of Policy Analysis and Information Systems*, Vol. 4, No. 2, pp. 125-161, 1980.

Michalski, R. S., Carbonell, J. and Mitchell, T. (Eds.), *Machine Learning: An Artificial Intelligence Approach*, TIOGA Publishing Company, Palo Alto, CA, 1983.

Michalski, R. S., Carbonell, J. and Mitchell, T. (Eds.), *Machine Learning: An Artificial Intelligence Approach, Vol II*, Morgan Kaufmann Publishers, Los Altos, CA, 1986.

Mitchell, T., Carbonell, J. and Michalski, R. S. (Eds.), *Machine Learning: Guide to Current Research,* Kluwer Publishing Co., 1986.

Mitchell, T. M., Keller, R. M. and Kedar-Cabelli, S. T., "Explanation-based Generalization: A Unifying View," *Machine Learning*, pp. 47- 80, Vol. 1, No. 1, 1986.

Rose, D. and Langley, P., "STAHLp: Belief Revision in Scientific Discovery," *AAAI-86, Fifth National Conference on Artificial Intelligence,* Vol. I, pp. 528-532, Philadelphia, PA, August 1986.

Winston, P. H., "Learning Structural Descriptions From Examples," Tech. Report AI TR-213, MIT, AI Lab, Cambridge, MA, 1977.

Winston, P. H., "Learning by Augmenting Rules and Accumulating Sensors," Chapter in *Machine Learning: An Artificial Intelligence Approach, Vol. II,* Morgan Kaufmann Publishers, Inc. 1986.

For AQ:

Michalski, R. S. and Larson, J. B., "INCREMENTAL GENERATION OF VL1 HYPOTHESES: The Underlying Methodology and the Description of Program AQ11," ISG 83-5, UIUCDCS-F-83-905, Department of Computer Science, University of Illinois, Urbana, IL, 1983.

Michalski, R. S., Mozetic, I., Hong, J. and Lavrac, N., "The AQ15 Inductive Learning System: An Overview and Experiments," Report No. UIUCDCS-R-86-1260, Department of Computer Science, University of Illinois, Urbana IL, July 1986.

Exemplary applications:

Michalski, R. S., Mozetic, I., Hong, J. and Lavrac, N.,"The Multi-Purpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains," *Proceedings of AAAI-86, Fifth National Conference on Artificial Intelligence,* Vol. 2, pp. 1041-1045, Philadelphia, PA, August 1986.

Mozetic, I., "Compression of the ECG Knowledge-base Using the AQ Induction Learning Algorithm", ISG 85-13, UIUCDCS-F-85-943, Department of Computer Science, University of Illinois, Urbana, IL, March 1985.

For INDUCE:

Bentrup, J., Mehler, G. and Riedesel, J., "INDUCE.4: A Program for Incrementally Learning Structural Descriptions from Examples", Technical Report UIUCDCS-F-87-958, Department of Computer Science, University of Illinois, Urbana, IL, 1987.

Hoff, W. A., Michalski, R. S. and Stepp, R. E., "INDUCE 2: A Program For Learning Structural Descriptions From Examples," Technical Report UIUCDCS-F-83-904, Department of Computer Science, University of Illinois, Urbana, IL, September, 1983.

Exemplary applications:

Lewis, C. M., "Identification of Rule-based Models," Report No. 86-5, Center for Man-Machine Systems Research, Georgia Institute of Technology, May 1986.

For CLUSTER:

Michalski, R. S. and Stepp, R. E., "Automated Construction of Classifications: Conceptual Clustering Versus Numerical Taxonomy," IEEE Transactions on Pattern Analysis and Machine Intelligence, 1983.

Michalski, R. S., and Stepp, R. E., "Clustering," *Encyclopedia of Artificial Intelligence*, S. Shapiro ed. John Wiley and Sons Publishers, New York, NY 1987, pp. 103-111.

Michalski, R. S., Stepp, R. E. and Diday, E., "A Recent Advance in Data Analysis: Clustering Objects into Classes Characterized by Conjunctive Concepts," in *Progress in Pattern Recognition*, Vol. 1, L. N. Kanall and A. Rosenfeld (Eds.), New York: North-Holland, pp. 33-56, 1981.

Stepp, R. E., "Conjunctive Conceptual Clustering: A Methodology and Experimentation," Ph.D. Thesis, Department of Computer Science, University of Illinois, Urbana IL, June 1984.

For SPARC:

Abbott, R., "The new Eleusis", available from Abbott at Box 1175, General Post Office, New York, NY 10001 ($1.00).

Dietterich, T. and Michalski, R. S., "Learning to Predict Sequences," Chapter in *Machine Learning: An Artificial Intelligence Approach Vol. II*, R. S. Michalski, J. Carbonell and T. Mitchell (Eds.), Morgan Kaufmann Publishers, Los Altos, CA, pp. 63-106, 1986.

Michalski, R. S., Ko, H. and Chen, K., "SPARC/E(V.2), An Eleusis Rule Generator and Game Player," ISG 85-11, UIUCDCS-F-85-941, Department of Computer Science, University of Illinois, Urbana, IL, February 1985.

Michalski, R. S., Ko, H. and Chen, K., "Qualitative Process Prediction: A Method and Program SPARC/G," *Expert Systems*, C. Guetler, (Ed.), Academic Press Inc., London, 1986.

For ABACUS:

Falkenhainer, B., "ABACUS: Adding Domain Constraints to Quantitative Scientific Discovery," ISG 84-7, UIUCDCS-F-84-927, Department of Computer Science, University of Illinois, Urbana, November 1984.

Falkenhainer, B. and Michalski, R.S., "Integrating Quantitative and Qualitative Discovery: The ABACUS System," in *Machine Learning: An Artificial Intelligence Approach, Volume III*, Kodratoff and Michalski, Eds., Morgan Kaufmann Publishers, San Mateo CA, 1990.

For EMERALD:

Kaufman, K., Schultz, A. and Michalski, R. S., "EMERALD 1: An Integrated System of

Machine Learning and Discovery Programs for Education and Research. Programmer's Guide for the Sun Workstation", *Reports of the Machine Learning and Inference Laboratory*, MLI 90-13, Center for Artificial Intelligence, George Mason University, Fairfax VA, 1990.

Michalski, R. S., "Machines That Learn and Discover", Artificial Intelligence Center, George Mason University, Fairfax VA, January 1988.