EMERALD 2:
An Integrated System of Machine Learning and Discovery Programs for Education and Research
User's Guide (Updated Edition)

Kenneth A. Kaufman
Ryszard S. Michalski

MLI 97-8
P 97-11
April, 1997

EMERALD 2: AN INTEGRATED SYSTEM OF MACHINE LEARNING AND DISCOVERY PROGRAMS FOR EDUCATION AND RESEARCH USER'S GUIDE (UPDATED EDITION)

Kenneth A. Kaufman and Ryszard S. Michalski

Machine Learning and Inference Laboratory, George Mason University,
Fairfax, VA 22030-4444

{kaufman, michalski}@mli.gmu.edu
http://www.mli.gmu.edu

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 U.S. 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. A new edition, under development, will be a C-based portable version of the system.

Acknowledgments

The EMERALD system integrates several machine learning and discovery programs that have been developed on the basis of research conducted by R.S. Michalski over the span of over twenty years, first at the University of Illinois at Urbana-Champaign, and then at the Machine Learning and Inference Laboratory at George Mason University. The Laboratory's research activities are currently supported in part by the National Science Foundation under Grants No. DMI-9496192 and IRI-9020266, in part by the Defense Advanced Research Projects Agency under Grant No. N00014-91-J-1351 administered by the Office of Naval Research, and in part by the Defense Advanced Research Projects Agency under Grants No.F49620-92-J-0549 and F49620-95-I0462, administered by the Air Force Office of Scientific Research.

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 Science Foundation under Grants No.MCS-74-03514, MCS-76-22940, MCS-79-06614, MCS-82-05166, MCS-83-06614, and NSF DSR-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 Advanced Research Projects Agency under grants administered by the Office of Naval Research No. N00014-85-K-0878, 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 original version of EMERALD that was presented at the exhibit "Robots and Beyond: The Age of Intelligent Machines," was provided by 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 Machine Learning and Inference Laboratory under the direction of R.S. Michalski, in collaboration with K. DeJong, K. Kaufman, A. Schultz, P. Stefanski and J. Zhang. We particularly acknowledge the contributions of A. Schultz, who implemented the port of EMERALD to the Sun workstation, and who contributed to the first version of this document. The authors also express gratitude to Dr. Murray Black, Dean of the School of Information Technology and Engineering, George Mason University, for the support to continue development of EMERALD and its adaptation to multiple platforms.

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. To serve these needs, two versions 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, Chicago, St. Paul and Columbus), and was presented at those museums during the years 1987-1989. The earlier version is now on permanent exhibition in Boston.

THE WORLD OF MACHINES THAT
LEARN AND DISCOVER
An Artificial Intelligence Exhibit
Presented by
GEORGE MASON UNIVERSITY
To begin the demonstration, press the SELECT button.

Figure 1: Welcome screen.

EMERALD (**E**xperimental **M**achine **E**xample-based **R**easoning **a**nd **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. A version of the system that runs on the Sun workstation is currently available, while a C-based portable system is currently under development.

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 a first step toward such a goal, the programs were integrated at the user interface level, i.e., the system allows the user easy access to each program through a menu, and facilitates the application of each to problems defined by the user, employing various predefined objects.

The capabilities of the EMERALD system include the ability to learn general concepts or decision rules from examples, create meaningful classifications of observations, predict sequences of objects, and discover 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, playing cards, etc. – so that they can be easily understood. These learning programs, however, have already been applied to and have the potential to be very 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 the 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 basic use and installation of the EMERALD 2 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 Common Lisp on a Sun workstation. It requires OpenWindows software and a monitor with full color graphics. A DecTalk voice synthesis module is optional, but highly recommended. The latter device should be hooked into a serial port corresponding to /dev/ttya. 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 workstation, 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 a file called **clx-lisp**), and then tell the Lisp system to load *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 if the system is a Sun 3 or a Sun 4, whether to enable the talk facility, and whether to use the local system for display. Normally, the latter question should be responded to affirmatively, but this option allows for cases of networked Suns in which EMERALD could 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 this window to the front.

After these questions are answered, the display will go dark until loading is complete. When the loading is complete, an initial screen will automatically come up (Figure 1). 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 start 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 chapter 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 goal of the EMERALD project was to create an exhibit displaying a set of pages (slides) through which a user could progress in much the manner one pages through a book. To this end, the system was structured in much the same way one organizes a text. The system currently 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 the user to the robot's abilities)

You Challenge Robot (Allows the user to experiment with the robot)

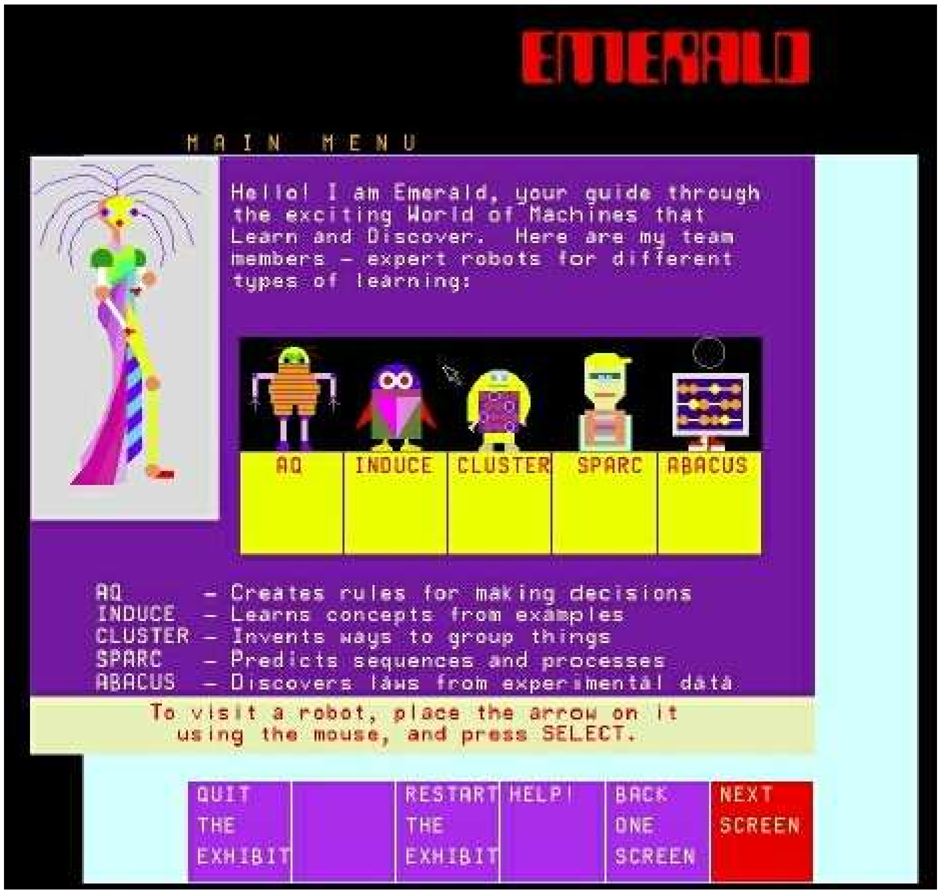
Find Out How Robot Works (Explains briefly the theory behind the robot's operation)

Different robots may have additional submodules. As the user progresses through the exhibit, two types of pages may be encountered, "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 the 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 that 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 (Figure 1) whenever:

- 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)

The initial screen displays the title of the demonstration with full-color graphics illustrating some of the icons encountered throughout execution. It also indicates, both in writing and by voice, how the user may proceed with the demonstration. It is in this state that EMERALD waits for a user. The first menu page represents the cover of the exhibit.



Hello! I am Emerald, your guide through the exciting World of Machines that Learn and Discover. Here are my team members - expert robots for different types of learning:

AQ	- Creates rules for making decisions
INDUCE	- Learns concepts from examples
CLUSTER	- Invents ways to group things
SPARC	- Predicts sequences and processes
ABACUS	- Discovers laws from experimental data

To visit a robot, place the arrow on it using the mouse, and press SELECT.

Figure 2: Main Menu.

The next page of the exhibit (Figure 2), labeled MAIN MENU, displays the contents of the exhibit. This page contains an image of EMERALD and each of the subordinate robots that the user can visit. On this page, the user first encounters the set of standard 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 other pages, 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 those robots, or may choose a standard option 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

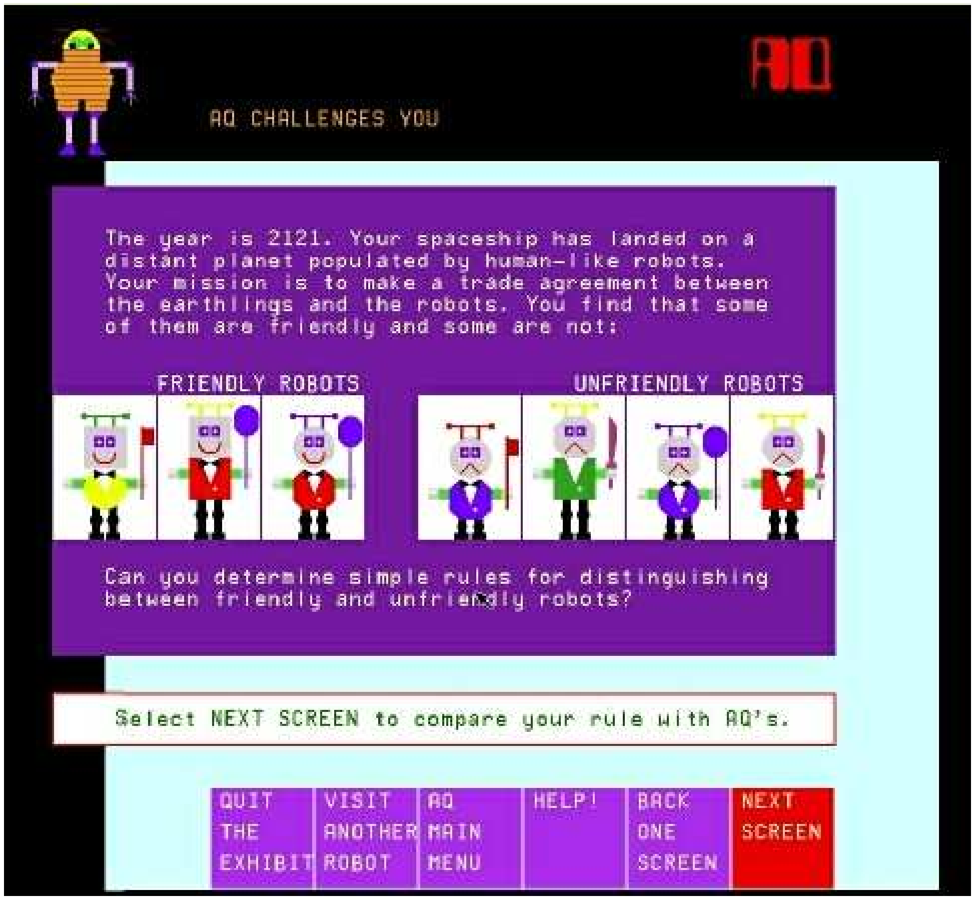
Upon entering the chapter describing AQ, one first encounters a menu showing the selection among the three standard subsections:

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 an advanced section at the top level. As the system is expanded, this division will eventually be made after the selection of the You Challenge AQ option.



The year is 2121. Your spaceship has landed on a distant planet populated by human-like robots. Your mission is to make a trade agreement between the earthlings and the robots. You find that some of them are friendly and some are not:

Can you determine simple rules for distinguishing between friendly and unfriendly robots?

Select NEXT SCREEN to compare your rule with AQ's.

Figure 3: AQ challenges you.

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

3.1.1 Option: AQ Challenges You

This section primarily contains seven menu pages – three pairs of problem/solution pages, and a summary page. Each of the three problem/solution pairs consists of a problem that challenges the user, followed by a page giving a solution to the problem. All problems are of a similar form, asking the user to find a rule that properly distinguishes between friendly and unfriendly robots (Figure 3). 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 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

The simple problem option 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 (Figure 4). It prompts the user to form two sets of robots for which AQ will learn a distinguishing rule. The two sets are introduced intuitively as members and non-members of a club that the user must define. The user may then choose the yellow box labeled 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:

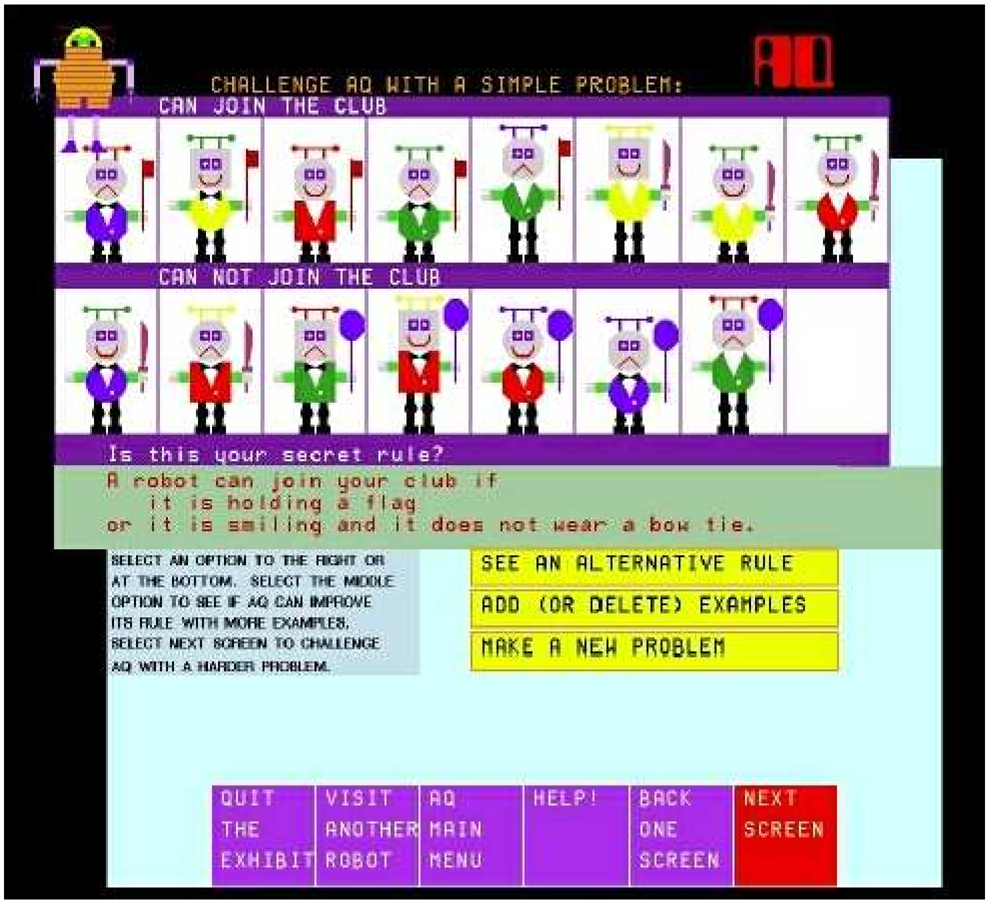
SEE AN ALTERNATIVE RULE	(display another rule to distinguish members from non-members)
ADD OR DELETE ROBOTS	(modify the current problem)
MAKE A NEW PROBLEM	(restart the work page)

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

COMPLEX PROBLEM

The complex problem option, 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 problem, except that there is room for up to four classes of robots. The page on which the rule is displayed, however, may contain two options not present in the simple problem section. They are:

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



Is this your secret rule?

A robot can join your club if
    it is holding a flag
or it is smiling and it does not wear a bow tie.

SELECT AN OPTION TO THE RIGHT OR AT THE BOTTOM. SELECT THE MIDDLE OPTION TO SEE IF AQ CAN IMPROVE ITS RULE WITH MORE EXAMPLES. SELECT NEXT SCREEN TO CHALLENGE AQ WITH A HARDER PROBLEM.

Figure 4: Challenge AQ with a problem.

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

3.1.3 Option: Find Out How AQ Works

This section consists of a single menu page. This 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. See the AQ-related references for more information on AQ15 and recent innovations in the AQ family of programs.

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.

EX2: How to distinguish groups of objects?
Distinguishes between multiple groups.

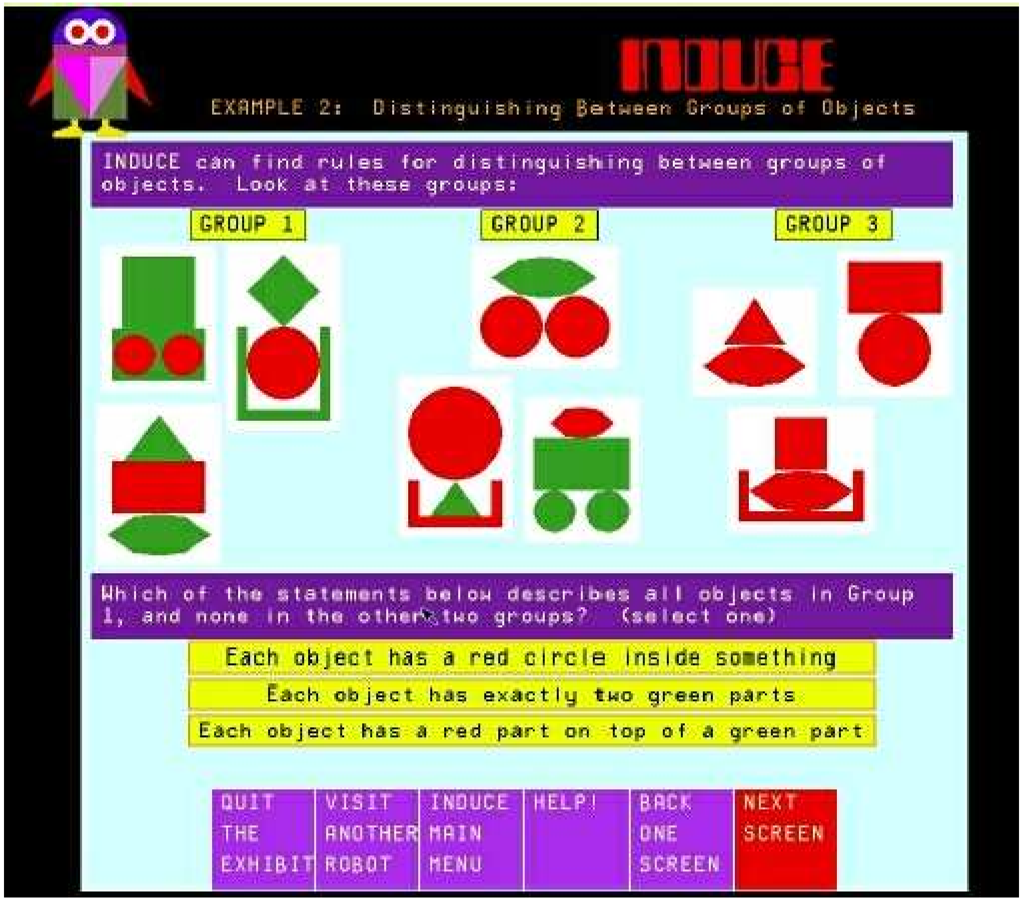
EX3: INDUCE discovers patterns in trains
Introduces the user to the domain in which INDUCE may be challenged.

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 that 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?



INDUCE can find rules for distinguishing between groups of objects. Look at these groups:

Which of the statements below describes all objects in Group 1, and none in the other two groups? (select one)

Each object has a red circle inside something

Each object has exactly two green parts

Each object has a red part on top of a green part

*Figure 5: INDUCE c*hallenges you.

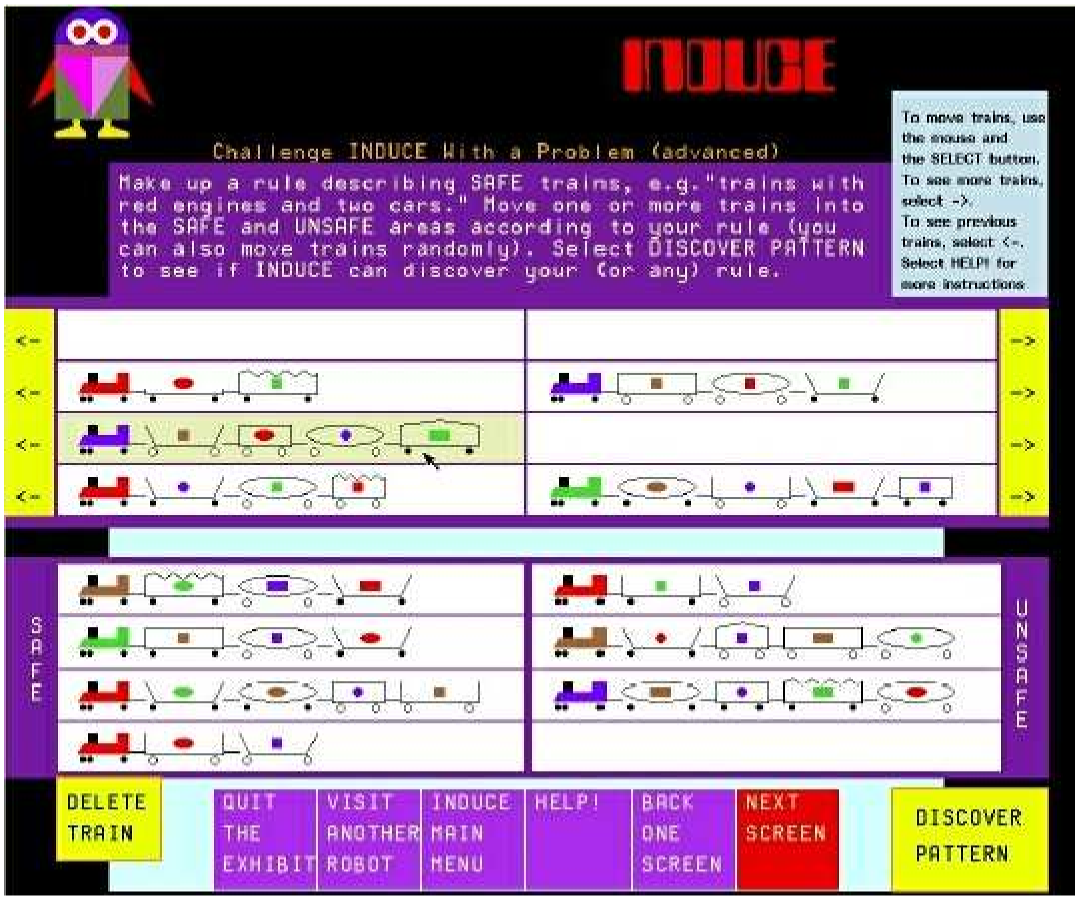
This example consists of a single work page challenging the user (Figure 5). The user is asked to devise a rule to distinguish between three sets of colored two-dimensional objects. Three possible rules are presented, and the user is asked to select one of these three possibilities. Upon selection of 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 that 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 (Figure 6). The page begins with a brief introduction, describing how to use the screen, followed by a white region representing a set of 48 trains.


INDUCE

Challenge INDUCE With a Problem (advanced)

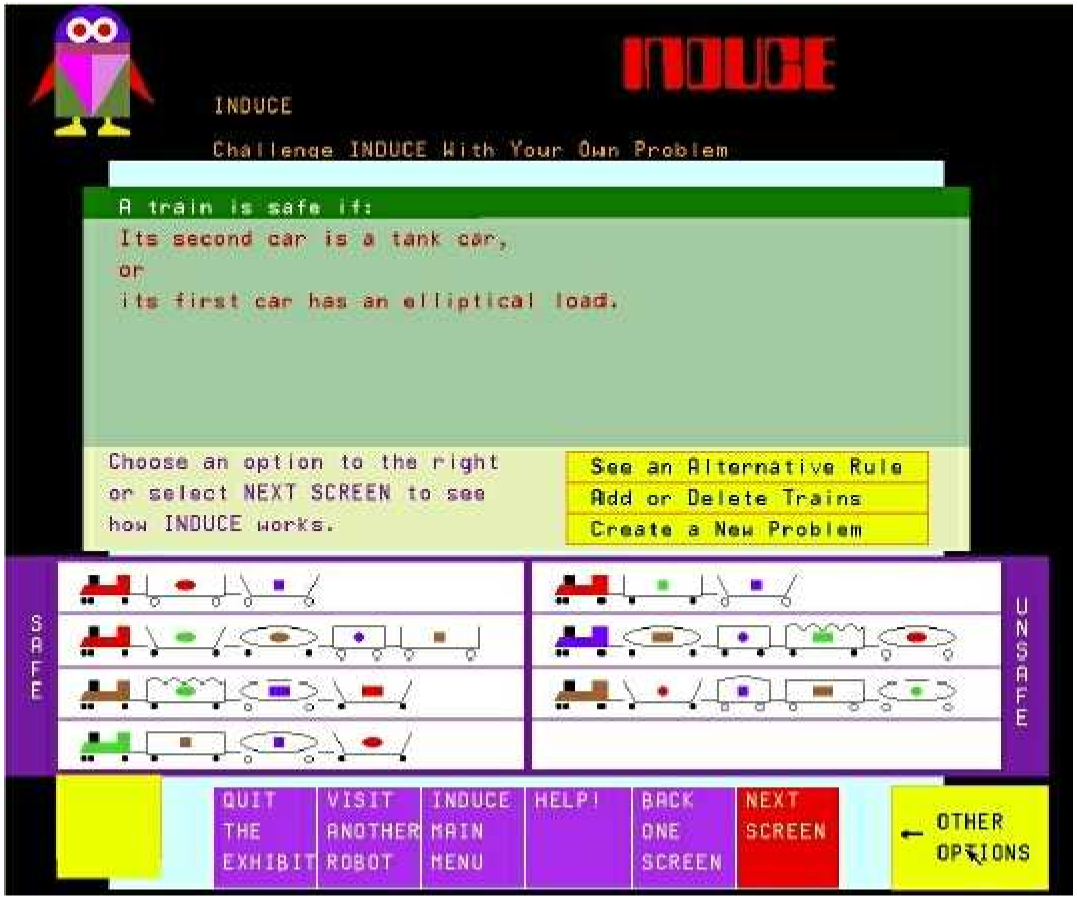
Make up a rule describing SAFE trains, e.g."trains with red engines and two cars." Move one or more trains into the SAFE and UNSAFE areas according to your rule (you can also move trains randomly). Select DISCOVER PATTERN to see if INDUCE can discover your (or any) rule.

To move trains, use the mouse and the SELECT button. To see more trains, select ->. To see previous trains, select <-. Select HELP! for more instructions.

Figure 6: Challenge INDUCE with a problem.

Because only eight 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 slots on 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 the user's making this choice, INDUCE devises a rule to describe SAFE trains. After presenting the devised rule in the area previously occupied by the set of trains (Figure 7), 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)



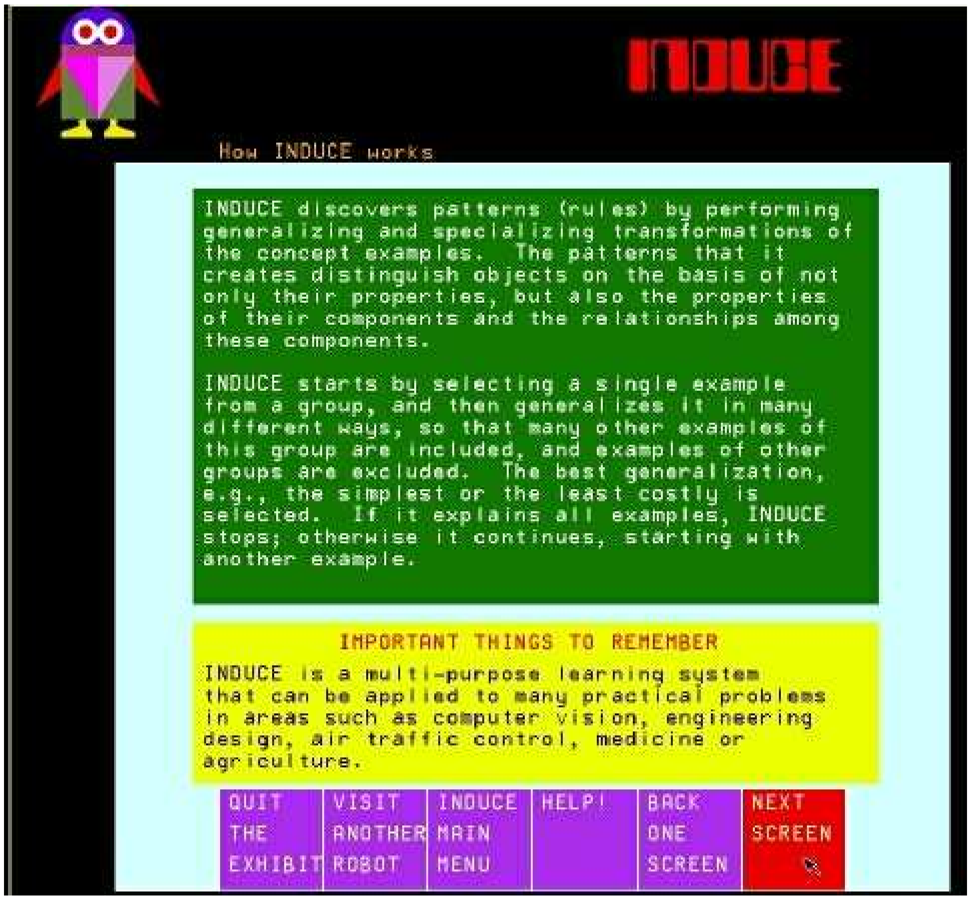
A train is safe if:
Its second car is a tank car,
or
its first car has an elliptical load.

Choose an option to the right or select NEXT SCREEN to see how INDUCE works.

Figure 7: INDUCE discovers a rule.

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 (Figure 8). See the INDUCE-related references for more information on the program.



INDUCE discovers patterns (rules) by performing generalizing and specializing transformations of the concept examples. The patterns that it creates distinguish objects on the basis of not only their properties, but also the properties of their components and the relationships among these components.

INDUCE starts by selecting a single example from a group, and then generalizes it in many different ways, so that many other examples of this group are included, and examples of other groups are excluded. The best generalization, e.g., the simplest or the least costly is selected. If it explains all examples, INDUCE stops; otherwise it continues, starting with another example.

IMPORTANT THINGS TO REMEMBER

INDUCE is a multi-purpose learning system that can be applied to many practical problems in areas such as computer vision, engineering design, air traffic control, medicine or agriculture.

Figure 8: INDUCE's capabilities.

3.3 CLUSTER Robot

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

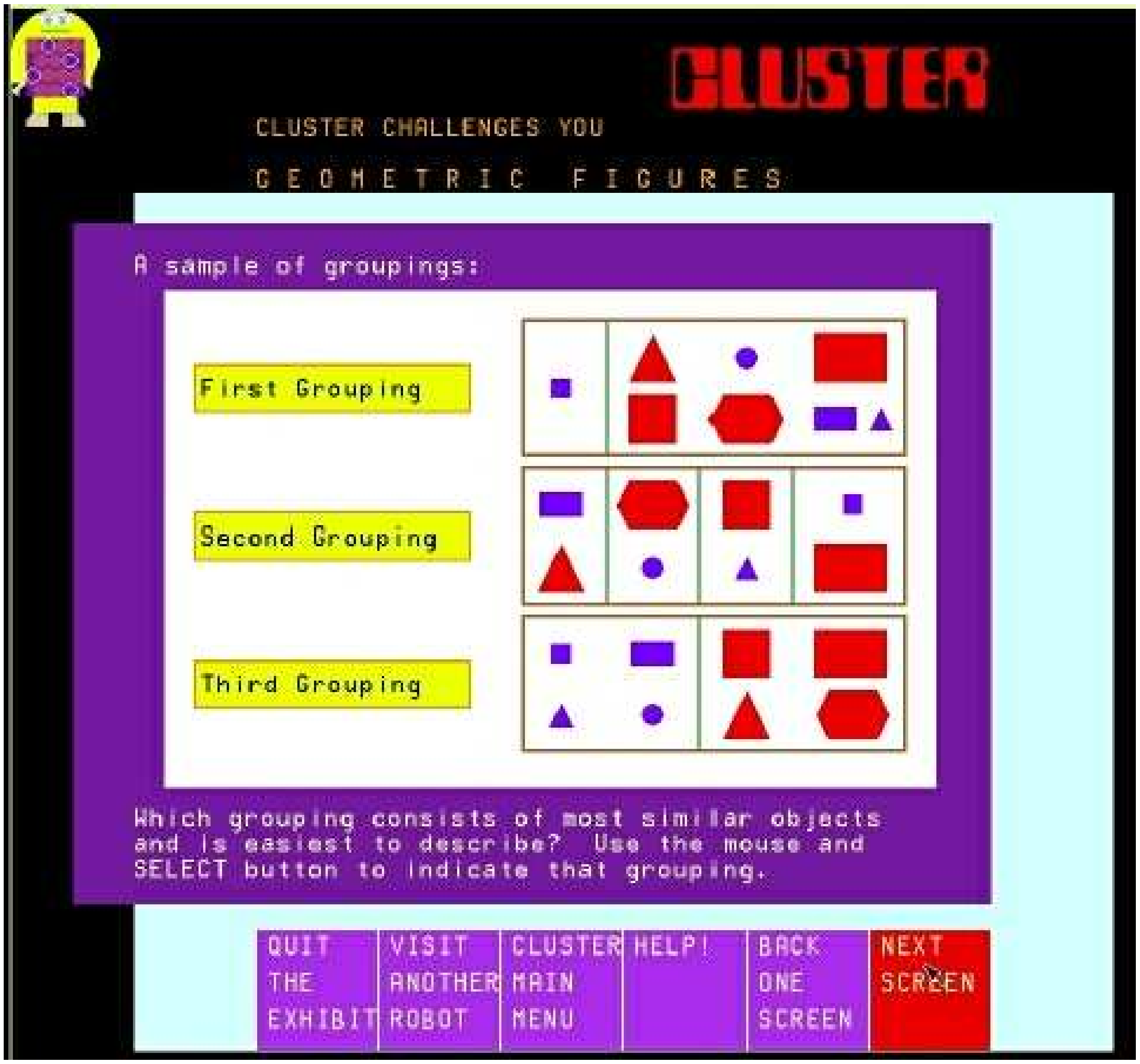
CLUSTER challenges you with a problem

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 (Figure 9). This page allows the user to select among three possible groupings of the eight objects. By placing the cursor controlled by the mouse over either a particular grouping or one of the yellow bars respectively labeled First Grouping, Second Grouping and 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 grouping agrees with that which CLUSTER would (hypothetically) select. If the user fails to select the correct clustering, the system allows the user to try again until the correct grouping is chosen. 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.

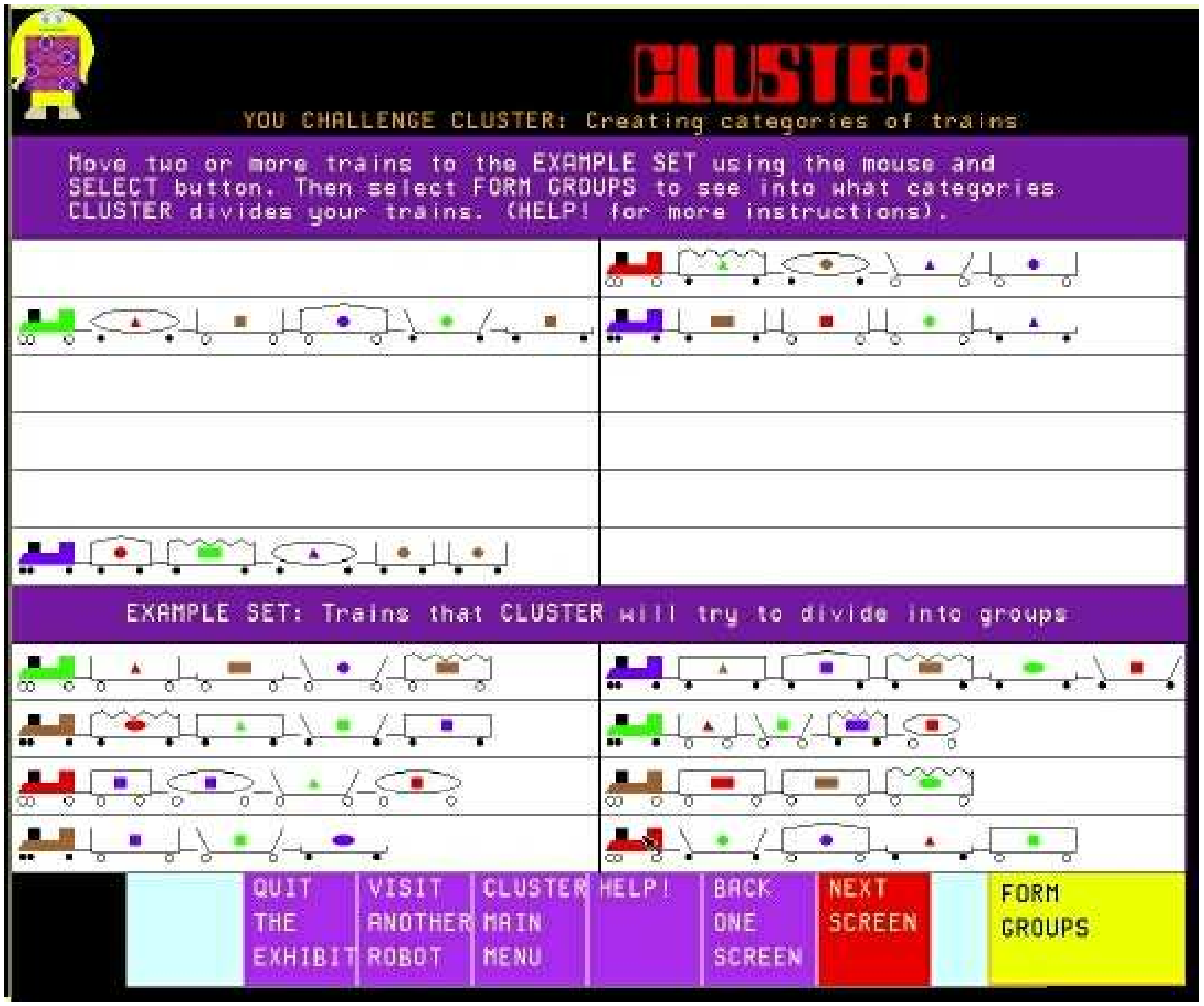


Which grouping consists of most similar objects and is easiest to describe? Use the mouse and SELECT button to indicate that grouping.

Figure 9: CLUSTER challenges you.

3.3.2 Option: You Challenge CLUSTER With A Problem

This section consists of a single work page (Figure 10). 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. Trains are moved by first using the mouse to place the selection cursor over a train, and pressing the select button, then moving the cursor to the destination white rectangle, and pressing the select button again. After more than one train has been placed in the EXAMPLE SET, the user may select the yellow region labeled FORM GROUPS. Upon this choice, CLUSTER begins attempting to divide the trains into groups based on properties such as the types of cars in the train and the loads they contain.



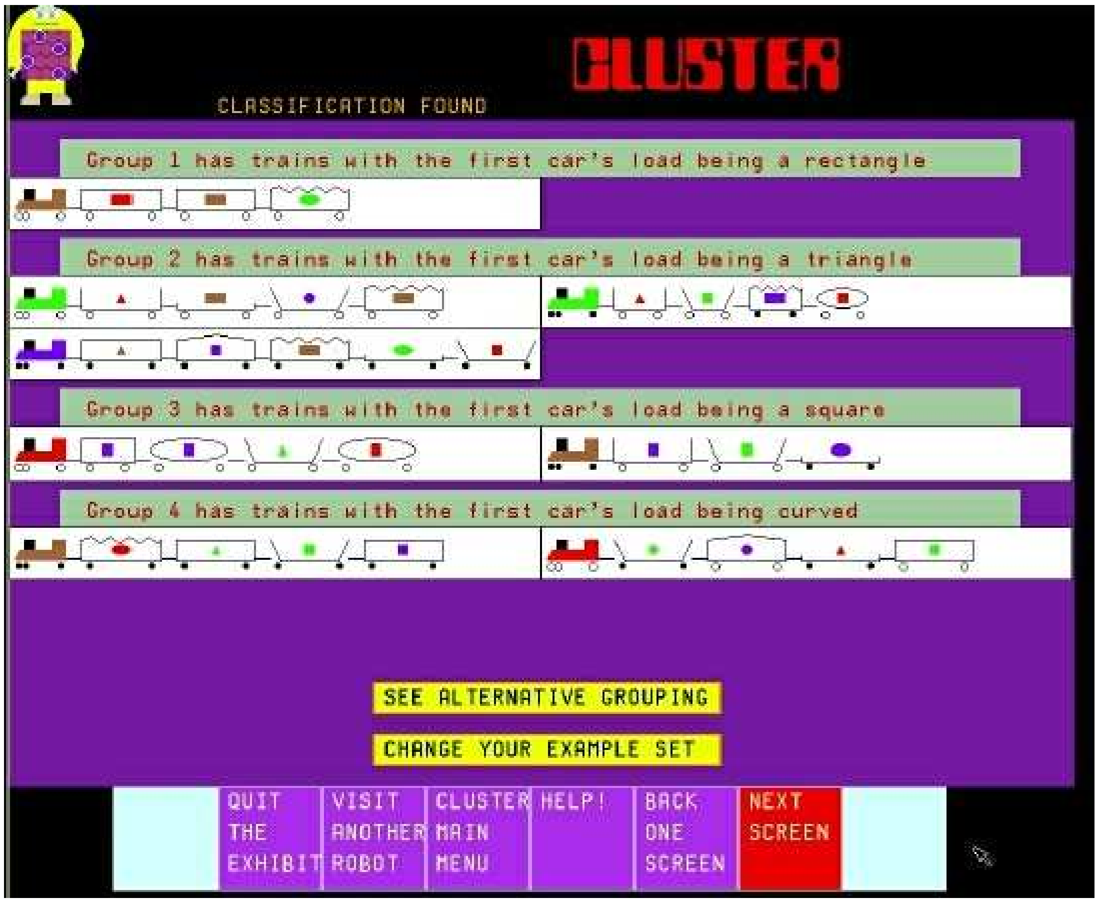
YOU CHALLENGE CLUSTER: Creating categories of trains

Move two or more trains to the EXAMPLE SET using the mouse and SELECT button. Then select FORM GROUPS to see into what categories CLUSTER divides your trains. (HELP! for more instructions).

EXAMPLE SET: Trains that CLUSTER will try to divide into groups

Figure 10: Challenge CLUSTER with a problem.

Upon classification, the groups formed are displayed at the top of the screen (Figure 11). Each group is described by a rule, shown in a green region (rule block), that 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 be displayed in the same manner as the first groups. If the second option, CHANGE YOUR EXAMPLE SET, is chosen, the screen is returned to the state it was in before FORM GROUPS was originally chosen.



Group 1 has trains with the first car's load being a rectangle

Group 2 has trains with the first car's load being a triangle

Group 3 has trains with the first car's load being a square

Group 4 has trains with the first car's load being curved

Figure 11: CLUSTER's classification.

3.3.3 Option: Find Out How CLUSTER Works

This section consists 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. See the CLUSTER-related references for more information on CLUSTER-2 and recent innovations to the CLUSTER family of programs.

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 page presents a problem that 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) that 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 is allowed to make another attempt. Upon the selection of 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 of the four select bars:

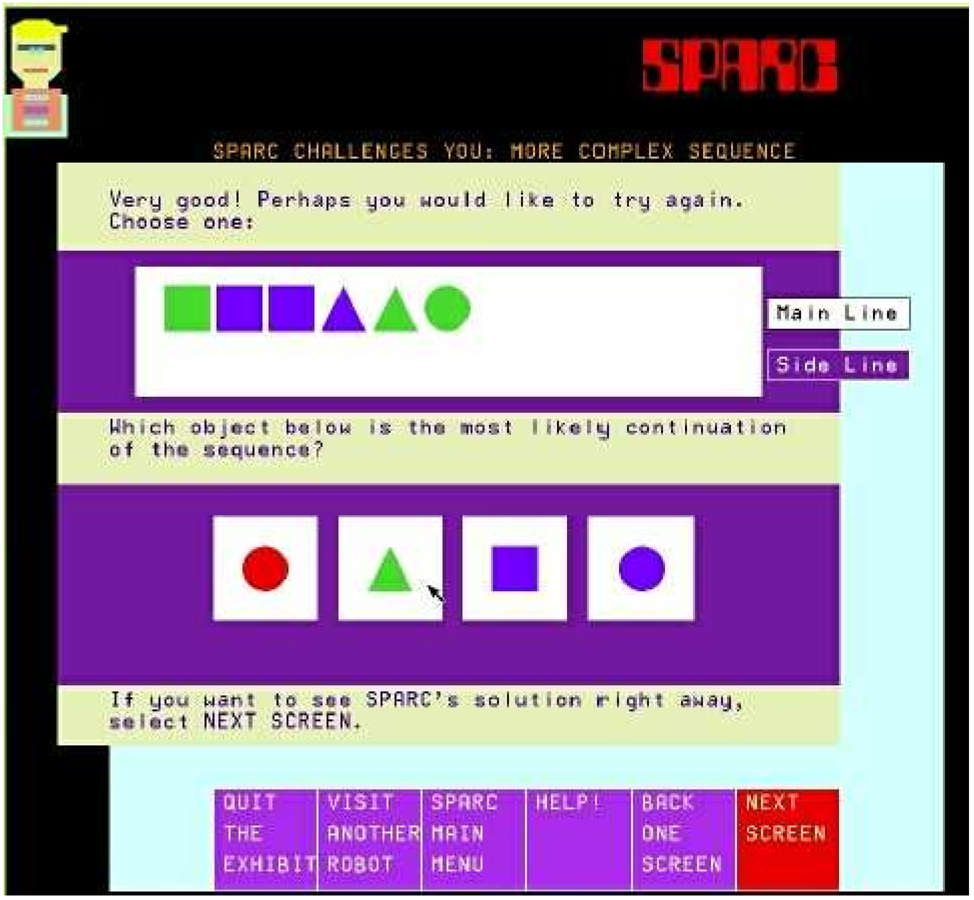
SPARC Challenges You: More Complex Sequence [geometric figures]

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 screen.



SPARC CHALLENGES YOU: MORE COMPLEX SEQUENCE

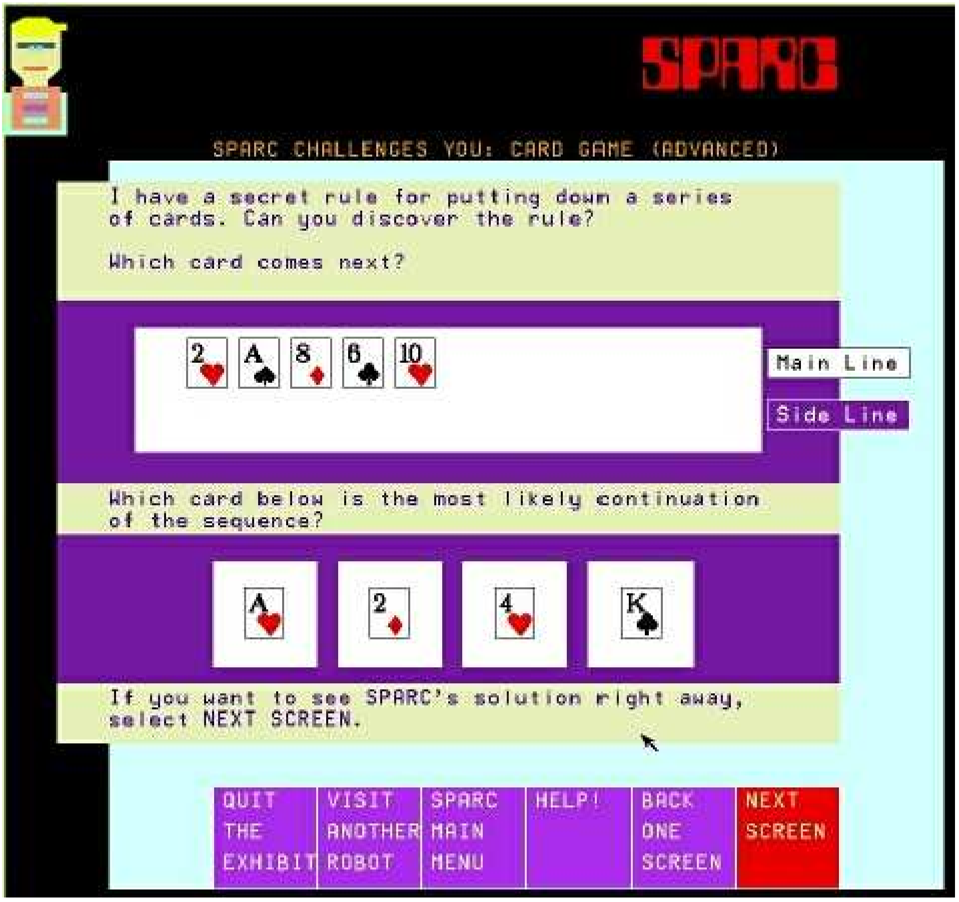
Very good! Perhaps you would like to try again.
Choose one:

Which object below is the most likely continuation of the sequence?

If you want to see SPARC's solution right away, select NEXT SCREEN.

Figure 12: SPARC challenges you with geometric figures.

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 (they 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 (Figures 12 and 13). 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 the 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, challenge SPARC with a problem, or go elsewhere via the standard options at the bottom of the screen.


SPARC CHALLENGES YOU: CARD GAME (ADVANCED)

I have a secret rule for putting down a series of cards. Can you discover the rule?

Which card comes next?

Which card below is the most likely continuation of the sequence?

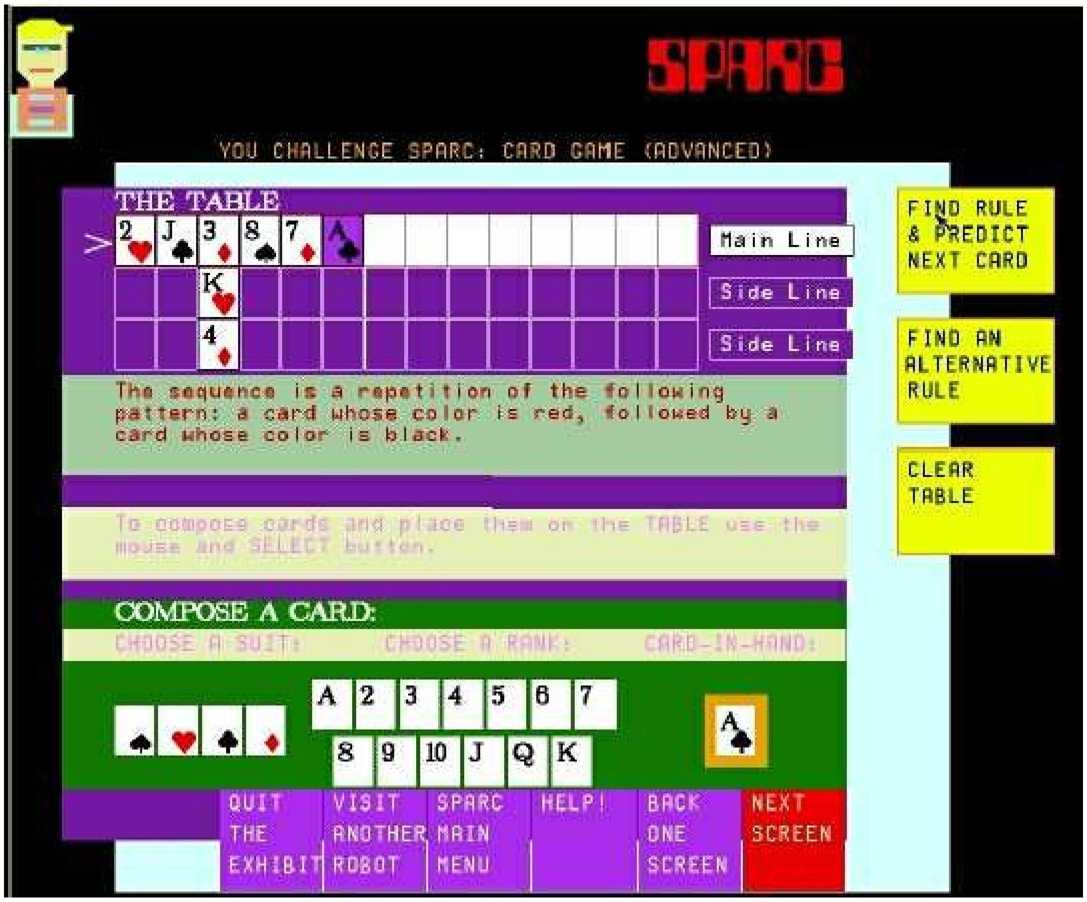
If you want to see SPARC's solution right away, select NEXT SCREEN.

Figure 13: SPARC challenges you with playing cards.

The Mined Channel example consists of 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 it safely. The code determines which paths are safe; other paths contain mines that must be avoided. Although not stated explicitly, the beacons have three features (shape, color, location along the path). The user is asked to determine from a part of the safe path shown on a map (presumably, an enemy was spotted negotiating a portion of the safe route) the next direction in which the ship should proceed. The choice is made by clicking the mouse on one of the arrows pointing 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 work 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 consists 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 sequence is a repetition of the following pattern: a card whose color is red, followed by a card whose color is black.

To compose cards and place them on the TABLE use the mouse and SELECT button.

Figure 14: You challenge SPARC.

The first time the user opens 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 be created using the “COMPOSE A CARD” option 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 user’s 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 Select button.) The user creates the sequence that SPARC is to continue in the top row. Negative examples are placed in a side line below the cards they are not allowed to follow. For example, if the fourth card in the sequence (the top row) may not be the three of hearts, that card could be placed under the third card in the row.

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

FIND RULE AND 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 (Figure 14). 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 not viewing it (the NO option). If the FIND AN ALTERNATIVE RULE option is chosen, a new rule is found explaining the current 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 the last element of the sequence, and extend the correct sequence by one card. Alternatively, cards can be placed below some earlier cards 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 to which SPARC may be applied. See the SPARC-related references for more information on the SPARC program.

3.5 ABACUS Robot

The layout of ABACUS differs from 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 about 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.

For further information on ABACUS and recent enhancements to the program, see the ABACUS-related references.

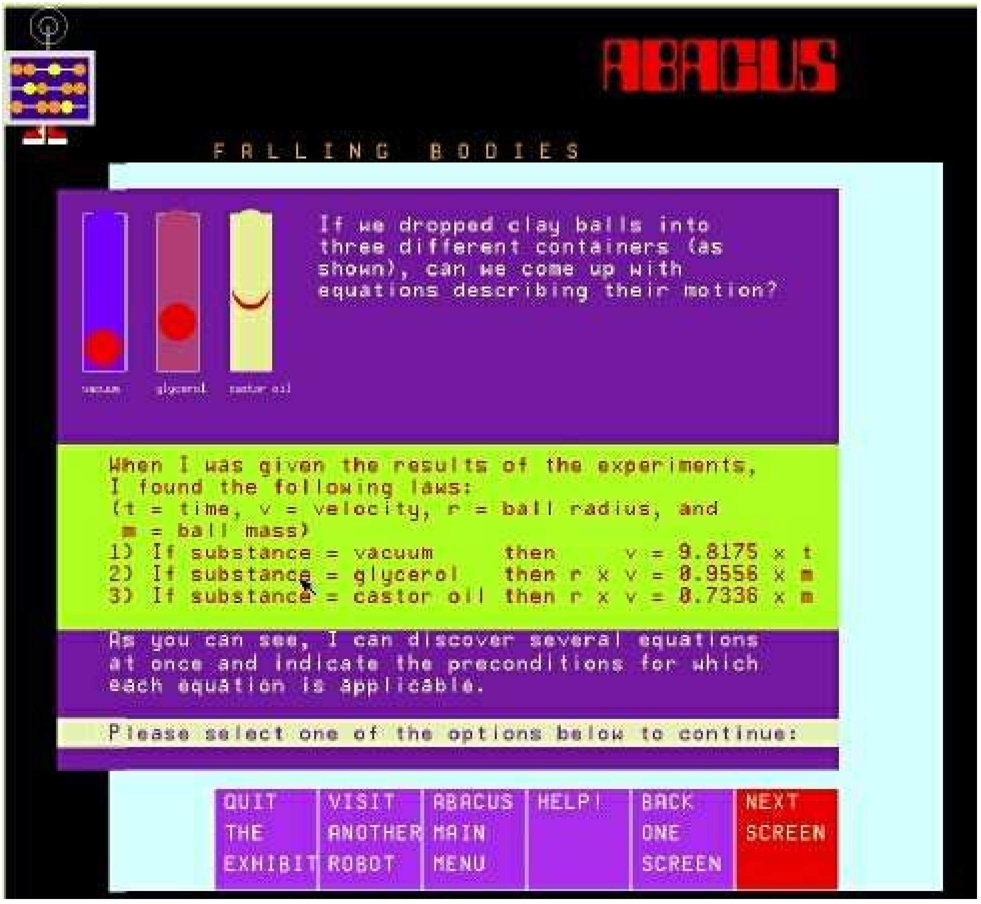
3.5.1 Option: Find Out About ABACUS

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

3 x R = V *2 x I = R* or *V = I x R*

The user is asked to select the rule that 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 that 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 (Figure 15), 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 form of rule may not be appropriate.



```
If we dropped clay balls into
three different containers (as
shown), can we come up with
equations describing their motion?

When I was given the results of the experiments,
I found the following laws:
(t = time, v = velocity, r = ball radius, and
 m = ball mass)
1) If substance = vacuum     then     v = 9.8175 x t
2) If substance = glycerol   then r x v = 0.9556 x m
3) If substance = castor oil then r x v = 0.7336 x m

As you can see, I can discover several equations
at once and indicate the preconditions for which
each equation is applicable.

Please select one of the options below to continue:
```

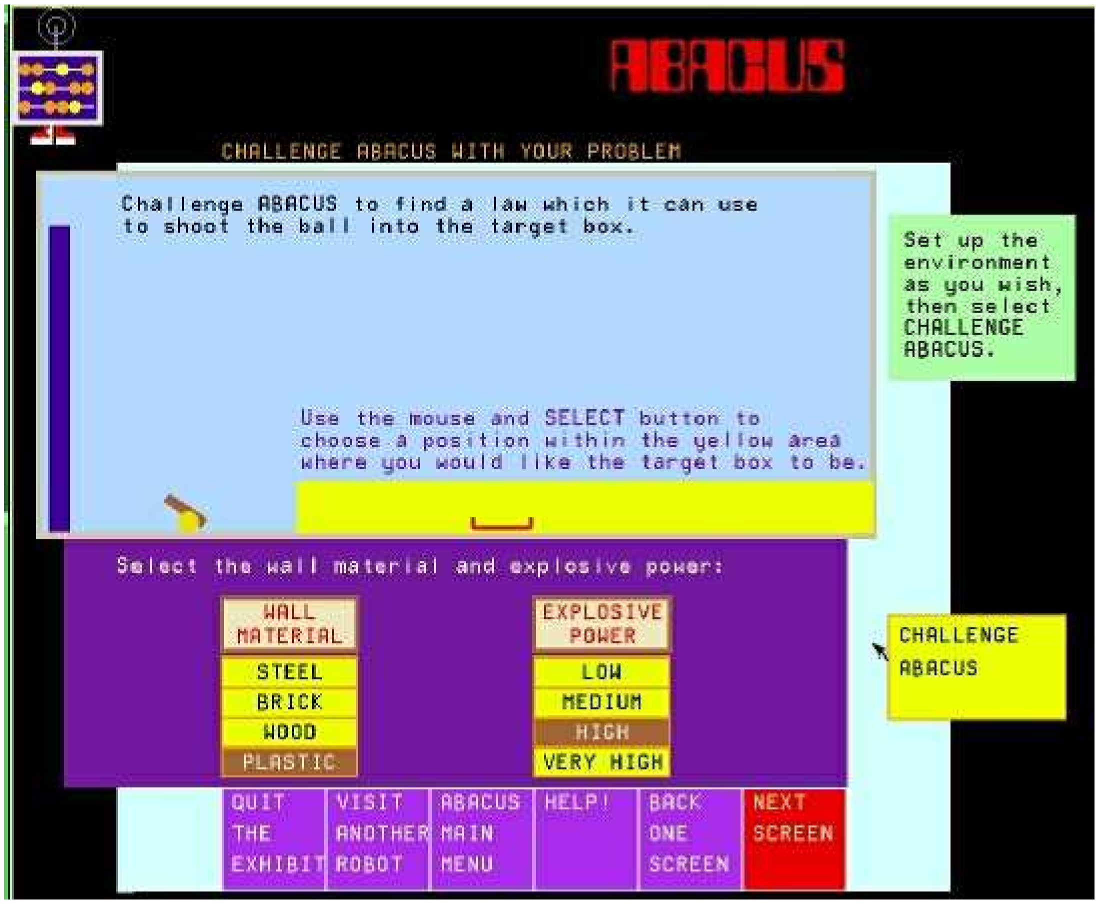
Figure 15: ABACUS discovers Stoke's Law.

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 this page, one attempts to aim a cannon so as to land a projectile in a box. The cannon fires a cannonball against a wall, and the ball rebounds in the direction of the box. Upon success, a menu page appears. On this page, the material of the wall and the speed at which the ball is fired are changed. At this point, the user may either choose to experiment in the new environment by selecting the yellow TRY AGAIN bar, or continue elsewhere by selecting one of the standard options. If the user continues to experiment, a new work page that reflects the changed environment appears. 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 two work pages and a menu page. The work pages (Figure 16) allow the user to set three parameters that define the environment. The user can position the box in which the cannonball must land, and set both the material of the wall (steel, brick, wood, or plastic, in descending order of elasticity) and the explosive power of the cannon (low, medium, high, or very high).



Challenge ABACUS to find a law which it can use to shoot the ball into the target box.

Set up the environment as you wish, then select CHALLENGE ABACUS.

Use the mouse and SELECT button to choose a position within the yellow area where you would like the target box to be.

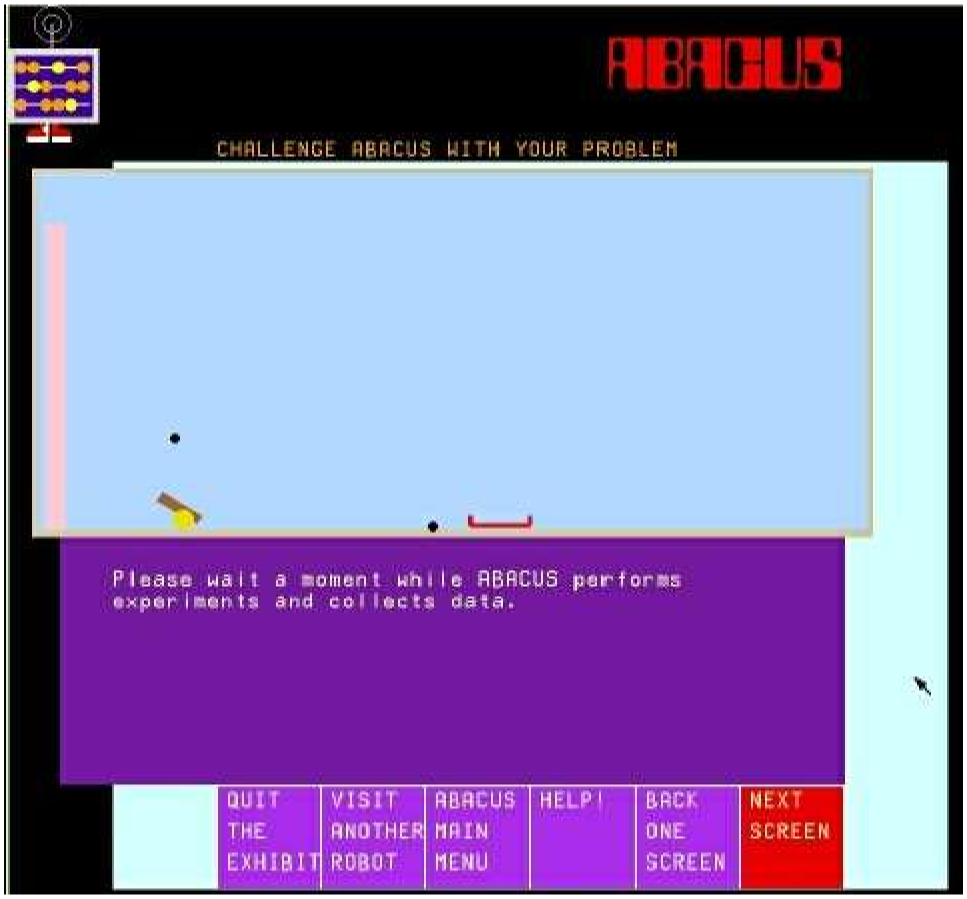
Select the wall material and explosive power:

WALL MATERIAL: [✓ PLASTIC]

EXPLOSIVE POWER: [✓ HIGH]

Figure 16: Setting up the environment.

Upon selection of the rectangle marked CHALLENGE ABACUS, ABACUS begins experimenting (Figure 17) in a trial-and-error manner analogous to what the user had to do in the previous section.



Please wait a moment while ABACUS performs experiments and collects data.

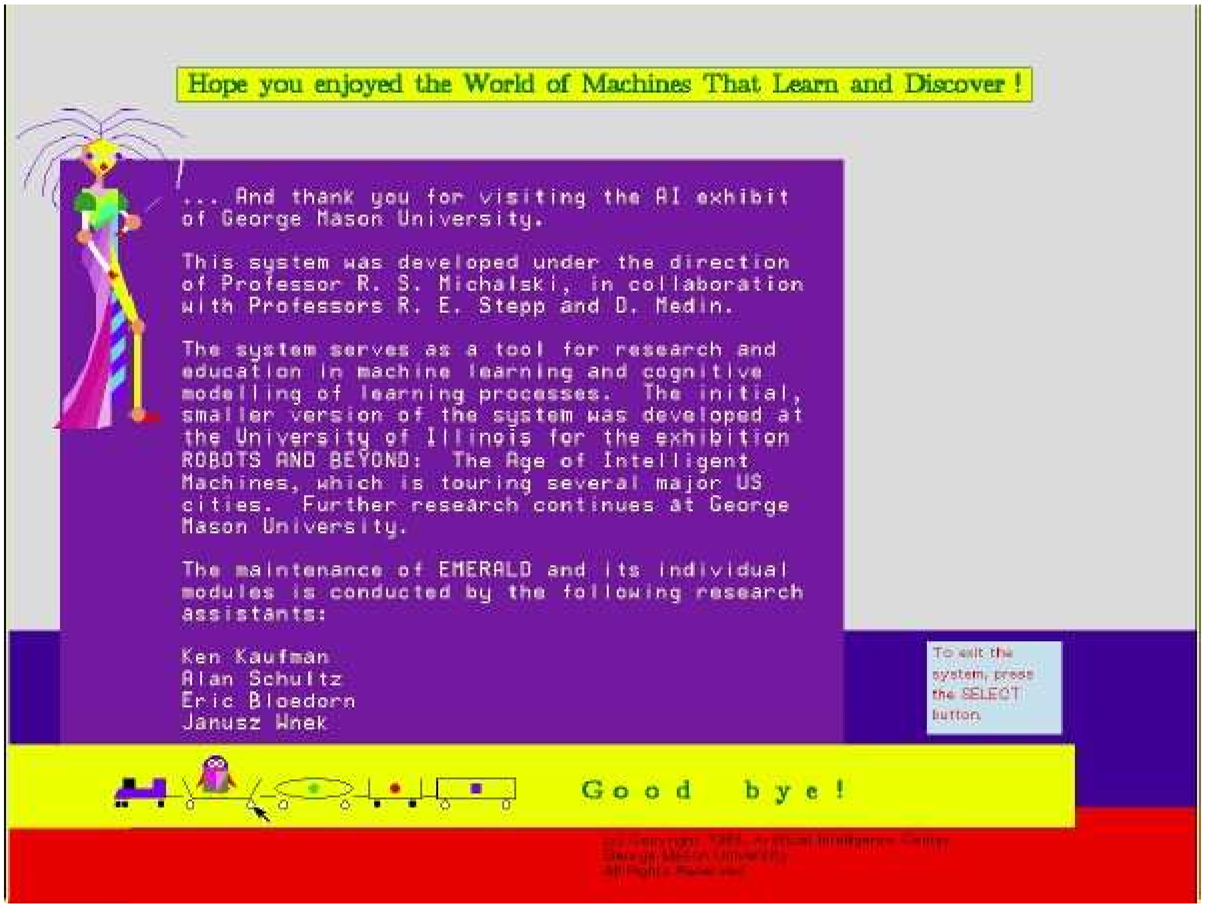
Figure 17: ABACUS conducts some experiments.

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 as placed, this is stated, and a new work page is displayed, allowing the user to reposition the box in a reachable location, and continue the simulation with the new position. The reachable area is indicated by a yellow region. The cannon is then fired several times, and the ball will land in the box each time. The use is now given two choices:

See The Equation Found by ABACUS

Challenge ABACUS With Another Environment

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



Hope you enjoyed the World of Machines That Learn and Discover !

... And thank you for visiting the AI exhibit of George Mason University.

This system was developed under the direction of Professor R. S. Michalski, in collaboration with Professors R. E. Stepp and D. Medin.

The system serves as a tool for research and education in machine learning and cognitive modelling of learning processes. The initial, smaller version of the system was developed at the University of Illinois for the exhibition ROBOTS AND BEYOND: The Age of Intelligent Machines, which is touring several major US cities. Further research continues at George Mason University.

The maintenance of EMERALD and its individual modules is conducted by the following research assistants:

Ken Kaufman
Alan Schultz
Eric Bloedorn
Janusz Wnek

To exit the system, press the SELECT button.

APPENDIX A INSTALLATION ON THE SUN WORKSTATION

Hardware requirements: A Sun workstation – Sun 3 or higher, with a color monitor. An 8 mm. 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:

```
lisp
(load “defsystem.lisp”)
(compile-clx)
(quit)
lisp
(load “defsystem.lisp”)
(load-system)
(disksave “clx-lisp”)
(quit)
```

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,” in Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (eds.), *Machine Learning: An Artificial Intelligence Approach*, Palo Alto: Tioga Publishing Company, pp. 41-82, 1983.

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

Lenat, D., “AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search,” Rept. STAN-CS-76-750, Computer Science Dept., 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, 1980.

Michalski R.S. (ed.), *Proceedings of the International Machine Learning Workshop,* University of Illinois Allerton House, Urbana, IL, 1983.

Michalski, R.S., “Concept Learning,” in Shapiro, S. (ed.), *Encyclopedia of Artificial Intelligence*, New York: John Wiley & Sons, pp. 185-194, 1987.

Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (eds.), *Machine Learning: An Artificial Intelligence Approach*, Palo Alto: Tioga Publishing Company, 1983.

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

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. and Kaufman, K., “Data Mining and Knowledge Discovery: A Review of Issues and a Multistrategy Approach,” in Michalski, R.S., Bratko, I. and Kubat, M. (eds.) *Machine Learning and Data Mining: Methods and Applications,*. London: John Wiley & Sons, 1998 (to appear).

Michalski, R.S. and Tecuci, G. (eds.), *Machine Learning: An Artificial Intelligence Approach, Vol. IV*, Los Altos, CA: Morgan Kaufmann Publishers, 1994.

Mitchell, T.M., Carbonell, J.G. and. Michalski, R.S. (eds.), *Machine Learning: A 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*, Vol. 1, No. 1, pp. 47-80, 1986.

[*] These References include papers describing algorithms implemented in EMERALD, as well as their extensions and recent developments.

Rose, D. and Langley, P., "STAHLp: Belief Revision in Scientific Discovery," *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86),* Philadelphia, PA, pp. 528-532, 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 Censors," in Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (eds.), *Machine Learning: An Artificial Intelligence Approach: Vol. II*, Los Altos, CA: Morgan Kaufmann Publishers, pp. 45-62, 1986.

For AQ:

Bloedorn, E. and Michalski, R.S., "The AQ17-DCI System for Data-Driven Constructive Induction and Its Application to the Analysis of World Economics," *Proceedings of the Ninth International Symposium on Methodologies for Intelligent Systems (ISMIS-96)*, Zakopane, Poland, 1996.

Bloedorn, E., Michalski, R.S. and Wnek, J., "Multistrategy Constructive Induction: AQ17-MCI," *Proceedings of the Second International Workshop on Multistrategy Learning (MSL93)*, Harpers Ferry, WV, pp. 188-203, 1993.

Michalski, R.S. and Imam, I.F., "On Learning Decision Structures," *Fundamenta Matematicae, 31(1), dedicated to the memory of Dr. Cecylia Raucher*, Polish Academy of Sciences, pp. 49-64, 1997.

Michalski, R.S. and Larson, J., "Incremental Generation of VL1 Hypotheses: The Underlying Methodology and the Description of Program AQ11," *Reports of the Intelligent Systems Group, ISG 83-5, UIUCDCS-F-83-905*, Department of Computer Science, University of Illinois, Urbana, January 1983.

Michalski, R.S., Mozetic, I., Hong, J. and Lavrac, N., "The AQ15 Inductive Learning System: An Overview and Experiments," *Reports of the Intelligent Systems Group, ISG 86-20, UIUCDCS-R-86-1260,* Department of Computer Science, University of Illinois, Urbana, 1986.

Wnek, J., Kaufman, K., Bloedorn, E. and Michalski, R.S., "Inductive Learning System AQ15c: The Method and User's Guide," *Reports of the Machine Learning and Inference Laboratory*, MLI 95-4, George Mason University, Fairfax, VA, 1995.

Wnek, J. and Michalski, R.S., "Hypothesis-driven Constructive Induction in AQ17-HCI: A Method and Experiments," *Machine Learning*, Vol. 14, No. 2, pp. 139-168, 1994.

Exemplary Applications:

Michalski, R.S., Kaufman, K. and Wnek, J., "The AQ Family of Learning Programs: A Review of Recent Developments and an Exemplary Application," *Reports of the Machine Learning and Inference Laboratory,* MLI 91-11, George Mason University, Fairfax, VA, 1991.

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 the Fifth National Conference on Artificial Intelligence, (AAAI-86),* Philadelphia, PA, 1986.

Mozetic, I., “Compression of the ECG Knowledge-base Using the AQ Inductive Learning Algorithm,” *Reports of the Intelligent Systems Group, ISG 85-13, UIUCDCS-F-85-943*, Department of Computer Science, University of Illinois, Urbana,1985.

For INDUCE:

Bentrup, J.A., Mehler, G.J. and Riedesel, J.D., “IINDUCE 4: A Program for Incrementally Learning Structural Descriptions from Examples,”" *Reports of the Intelligent Systems Group, ISG 87-2, UIUCDCS-F-87-958*, Department of Computer Science, University of Illinois, Urbana, 1987.

Hoff, W., Michalski, R.S. and Stepp, R., “INDUCE 2: A Program for Learning Structural Descriptions from Examples,” *Reports of the Intelligent Systems Group, ISG 83-4, UIUCDCS-F-83-904*, Department of Computer Science, University of Illinois, Urbana, 1983.

Exemplary Applications:

Bloedorn, E., Imam, I., Kaufman, K., Maloof, M., Michalski, R.S. and Wnek, J., “HOW DID AQ FACE THE EAST-WEST CHALLENGE? An Analysis of the AQ Family's Performance in the 2nd International Competition of Machine Learning Programs,” *Reports of the Machine Learning and Inference Laboratory*, MLI 95-3, George Mason University, Fairfax, VA, 1995.

Lewis, C.M., “Identification of Rule-Based Models,” Report No. 86-5, Center for Machine Systems Research, Georgia Institute of Technology, 1986.

For CLUSTER:

Fischthal, S., “A Description and User's Guide for CLUSTER/2C++ A Program for Conjunctive Conceptual Clustering,” *Reports of the Machine Learning and Inference Laboratory,* MLI 97-10, George Mason University, Fairfax, VA, 1997.

Michalski, R.S. and Stepp, R., “Automated Construction of Classifications: Conceptual Clustering versus Numerical Taxonomy,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 4, pp. 396-410, 1983.

Michalski, R.S. and Stepp, R., “Clustering,” in Shapiro, S. (ed.), *Encyclopedia of Artificial Intelligence*, New York: John Wiley & Sons, 1987.

Michalski, R.S., Stepp, R. and Diday, E., “A Recent Advance in Data Analysis: Clustering Objects into Classes Characterized by Conjunctive Concepts,” in Kanal, L. and Rosenfeld, A. (eds.), *Progress in Pattern Recognition, Vol. 1*, North-Holland, pp. 33-55, 1981.

Stepp, R., “Conjunctive Conceptual Clustering: A Methodology and Experimentation,” Ph.D Thesis, UIUCDCS-R-84-1189, Department of Computer Science, University of Illinois, Urbana, 1984.

For SPARC:

Abbott, R., “The New Eleusis,” available from the author at Box 1175, General Post Office, New York, NY 10001, 1977.

Dietterich, T. and Michalski, R.S., “Learning to Predict Sequences,” in Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (eds.), *Machine Learning: An Artificial Intelligence Approach: Vol. II*, Los Altos, CA: Morgan Kaufmann Publishers, pp. 63-10662, 1986.

Michalski, R.S., Ko, H. and Chen, K., “SPARC/E(V.2), An Eleusis Rule Generator and Game Player,” *Reports of the Intelligent Systems Group, ISG 85-11, UIUCDCS-F-85-941*, Department of Computer Science, University of Illinois, Urbana, 1985.

Michalski, R.S., Ko, H. and Chen, K., “Qualitative Prediction: The SPARC/G Methodology for Inductively Describing and Predicting Discrete Processes,” in *Expert Systems*, London: Academic Press Inc., 1986.

For ABACUS:

Falkenhainer, B., “ABACUS: Adding Domain Constraints to Quantitative Scientific Discovery,” *Reports of the Intelligent Systems Group, ISG 84-7, UIUCDCS-F-84-927*, Department of Computer Science, University of Illinois, Urbana, 1984.

Falkenhainer, B. and Michalski, R.S., “Integrating Quantitative and Qualitative Discovery in the ABACUS System,” in Kodratoff, Y. and Michalski, R.S. (eds.), *Machine Learning: An Artificial Intelligence Approach, Vol. III,* San Mateo, CA, pp. 153-190, Morgan Kaufmann Publishers, 1990.

Greene, G.H., “The Abacus.2 System for Quantitative Discovery: Using Dependencies to Discover Non-Linear Terms,” *Reports of the Machine Learning and Inference Laboratory*, MLI 88-4, George Mason University, Fairfax, VA, 1988.

Michael, J.B., “Validation, Verification, and Experimentation with Abacus2,” *Reports of the Machine Learning and Inference Laboratory*, MLI 91-8, George Mason University, Fairfax, VA, 1991.

For EMERALD:

Kaufman, K. and Michalski, R.S., “EMERALD 2: An Integrated System of Machine Learning and Discovery Programs to Support Education and Experimental Research,” *Reports of the Machine Learning and Inference Laboratory*, MLI 93-10, George Mason University, Fairfax, VA, 1993.

Kaufman, K. and Michalski, R.S., “EMERALD 2: An Integrated System of Machine Learning and Discovery Programs for Education and Research, Programmer's Guide for the Sun Workstation (Updated Edition),” *Reports of the Machine Learning and Inference Laboratory*, MLI 97-9, George Mason University, Fairfax, VA, 1997.

Michalski, R.S., “Machines that Learn and Discover,” Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA, 1988.

A publication of the *Machine Learning and Inference Laboratory*
School of Computational Sciences
George Mason University
Fairfax, VA 22030-4444 U.S.A.
http://www.mli.gmu.edu

Editor: R. S. Michalski
Assistant Editor: K. A. Kaufman

The *Machine Learning and Inference (MLI) Laboratory Reports* are an official publication of the Machine Learning and Inference Laboratory, which has been published continuously since 1971 by R.S. Michalski's research group (until 1987, while the group was at the University of Illinois, they were called ISG (Intelligent Systems Group) Reports, or were part of the Department of Computer Science Reports).