84-9

ABACUS

Adding Domain Constraints to Quantitative Scientific Discovery

Brian Falkenhainer

November, 1984
Department of Computer Science
University of Illinois
Urbana, Illinois 61801

ISG 84-7

File No. UIUC-DCS-F-84-927

This work was supported in part by the National Science Foundation under grant NSF DCR 84-06801 and by the Office of Naval Research under grant no. N00014-82-K-0186.

Acknowledgements

Thanks go to Professor Michalski for his support on this project and for his many helpful suggestions. I would also like to thank Ken Forbus for offering helpful comments on this paper and for ideas on the future course of the project. I also wish to express my gratitude to the members of ISG (namely Tony Nowicki and Jeff Becker) who spent the time to proof-read this document. Finally, I am grateful to Kaihu Chen for providing me with sample input data which caused me to improve the AB algorithm.

Table of Contents

ABACUS:
Adding Domain Constraints to Quantitative Scientific Discovery

Abstract

This paper describes the ABACUS quantitative scientific discovery system. ABACUS discovers mathematical relationships from a given set of numerical and nominal valued events. The system is able to subdivide these events into classes in terms of a mathematical formula which holds for a given class. ABACUS then generates descriminant descriptions of the classes, providing a domain constraint on each formula. The ABACUS algorithms are presented here (AB and Aq) and some of the results are shown.

1. Introduction

In recent years there have been a number of programs and theories investigating learning by discovery in the scientific domain. Some are designed so that they may begin with only a few simple facts and others, most notably the BACON series [Langley 79,81,83,84], start with raw experimental data and attempt to discover an empirical law which summarizes the observed data. Many of these types of programs suffer from imposing certain requirements upon the data. Often the user must initially specify which variables to treat as independent and which to treat as dependent. The programs are limited to proposing only one relation (or more recently several alternate relations [Langley 84]) as the law which describes the given data. Thus, the given data should not represent a mixing of relations. In addition, these types of programs are unable to specify the conditions under which the discovered relations hold. In scientific discovery, for example, part of the problem is discovering the constraints under which the newly found theorem holds. When taught to use Newton's laws of motion, we are cautioned that they are only accurate for objects traveling at speeds relatively slow when compared to the speed of light. In attempting to classify large amounts of data, we may wish to summarize the individual classes in terms of mathematical formulae which hold for a given class. A distinction must then be made between the classes to prevent these formulae from being used when not applicable. In other words, it would be very helpful if the domain of the discovered relation was stated along with the relation itself.

In response to these problems and needs, I have constructed a new system which I call ABACUS. Like many of its predecessors, ABACUS is initially presented with a collection of raw data. Each row in the data is designated as an "event" which represents the values of all the variables in a single observation. Thus, if the given variables are A, B, and C, an event might be given as 1, 2, and 4 representing the values for A, B, and C respectively. No other information need be given, except for declaring each variable as being either numerical or nominal. The search path is chosen by the program using various heuristics which take into account what has been discovered so far. When a relation is found which describes a portion of the data, the events covered by this relation are removed, placed in a unique set, and the process starts over to search for new relations to describe the remaining events. Once all the events are so analyzed, the resulting sets of events are examined to discover the constraints which describe and distinguish these groupings.

ABACUS

Let us examine a very simple ABACUS session. The data being investigated in this example has been plotted in Fig. 1. Imagine that this plot represents the observed values of the current, I, and the voltage, V, for some unusual device in an electrical circuit.

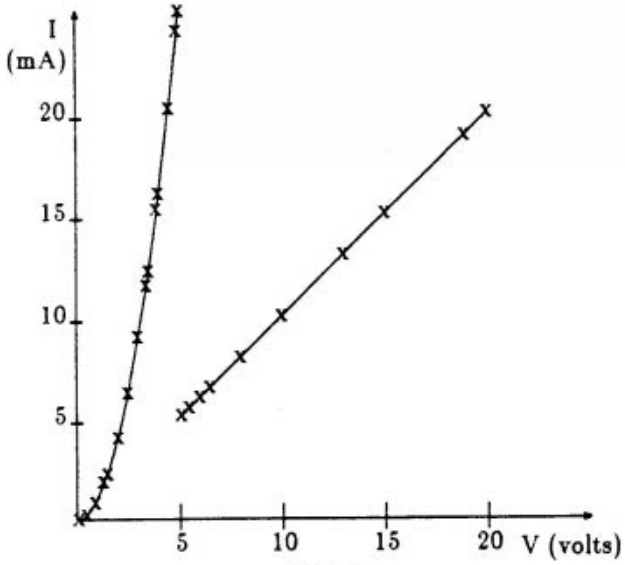

Fig. 1.

ABACUS will first sort the data so that patterns may be easily detected. For this example, the program will sort all of the values of I and, for sets of events where I takes on the same values, it will sort the corresponding values of V. Simple relation finding routines will then take over and notice that when I increases, V also increases. This relation holds for a high enough percentage of the events (100% here) to justify the creation of a new variable whose values will be calculated from V/I. This new variable both provides a summarization of the variables it represents and acts as a separate variable itself. The values of the new variable are now examined to see how many of them take on the same value. It is noticed that for approximately 50% of the events, the new variable (V/I) is equal to 1. Since this represents a large portion of the data, this variable is marked for later reference. The above process now continues in search of a higher percentage of constancy. For the data being examined here, the program fails to find anything better and so settles for this first generated variable and the (V/I = 1) relation. The events satisfying this relation are now removed from further consideration and grouped into a set designated "class-a". The program now starts over on the remaining data and once again generates as the first new variable V/I. For these events, however, no constant value is found to hold and so the routine must continue on. The new variable is now compared to the values for I and it is found that whenever I goes up, so do the values for the V/I variable. A new variable is thus generated which takes on the values (V/I) / I. It is then found that for all of the events the newest variable is equal to 1. Having discovered a relation which holds for all of the current data, the search algorithm ends

immediately with this finding and generates a new set designated "class-b". Because all of the events have now been successfully described, the first phase of the process is complete. Now the program attempts to discover what factors made class-a different from class-b (a different range of values for V) by using an algorithm known as Aq. This is an algorithm which, given event sets such as "class-a" and "class-b", will generate a description for each class which covers the examples of that class and none of the examples in any other class. This is called a descriminant description in that it may be used to determine to which class a given event belongs. The details of the Aq algorithm will be discussed later and for now the final results for this simple example are shown (Fig. 2.):

```
  a-outhypo
IF
     1    [V= 5.100.. 20.000]
THEN
     I = V
```

```
  b-outhypo
IF
     1    [V= 0.100.. 5.000]
THEN
     I = V^2
```

Fig. 2.

1.1. Viewpoint

At this point it should be noted that all of the examples and displayed output given in this paper are from actual executions of the ABACUS Pascal program. Data for all of the examples was generated by hand in order to save time and, where pointed out, data discrepancies and "noise" were also introduced by hand in order to demonstrate some of the powers of the system. In the future I would like to gather examples from real experimental situations to observe what effect this may have on results. My current opinion is that this should not create any problems.

2. Architecture

ABACUS uses a basic two-stage methodology. First, the given data is examined for empirical rules which hold over some subset of the events. As each empirical rule is discovered, the events which conform to this rule are removed entirely from the data to form a new class set. This process continues until all of the original data is removed in the form of class sets or until no more rules can be found. In the later case, the remaining events are gathered up into a miscellaneous class set. Once all of the data has been separated into disjoint classes, the second phase of the process begins. This stage applies the Aq algorithm to the classes to create discriminate descriptions of these event sets. With the descriptions and the previously discovered empirical rules which describe each class, we now have the derived if-then rules which were shown above.

2.1. AB Process

AB is the term used to describe the first stage of the over-all ABACUS system. Once the input data has been read and properly organized, the AB routines are called to search for

empirical rules and separate the data into disjoint classes according to these discovered rules.

2.1.1. Search

The basic methodology employed by AB is a depth-first search through the useful permutations of variable relations. At each level or node, all permutations of existing variables (both those from the original input data and those generated previously) are examined for a possible relationship. If a relation is found, the program advances on to the next level of description, adding the new resulting variable to its list of variables. Because of its ability to backtrack, the relation finding routine can be made simple and new levels of description may be generated without placing too much emphasis on verifying if the chosen path is the best one to take. If a level of description is found to take the program along a non-terminating path, the program backs up to the previous level and continues on, removing this path from present consideration. All paths and permutations of variable pairs are checked in this manner until either a terminating condition is reached or the entire search space has been explored. An algorithm parameter specifies the allowed search depth and the search space and non-terminating paths are defined in terms of this value.

ABACUS's ability to discover laws when irrelevant variables are present is heavily intertwined with its backtracking capabilities. An irrelevant variable may display some apparent relation to another variable and a new level of description is generated relating these two variables. Because the program has entered a new level with an irrelevant variable as one of its components, no terminating condition will be reached as it proceeds on to higher levels. Backtracking, however, enables the program to recover from this false path and proceed on in another direction. Such forward and backward movement will continue until a final terminating condition is found. Notice that with this design, it will be able to locate the proper relation even when many irrelevant and interfering variables are present in the data.

As an example, let us look at how ABACUS is able to discover a law describing the speed of sound in air. This law, as traditionally presented in the physics texts, states that the speeds V1 and V2 of sound through air at absolute temperatures (K) T1 and T2 are related by the formula:

$$\frac{V1}{V2} = \left(\frac{T1}{T2}\right)^{1/2}$$

More simply stated, any sound velocity and temperature are related by $V^2 / T = 401.322$. The speed of sound is essentially independent of pressure, frequency, and wavelength. In this example, ABACUS is presented with sets of data giving values for velocity (V), temperature (T), pressure (P), and frequency (f). On its first pass through the data, ABACUS discovers a loose relationship between velocity and frequency and thus generates the new variable ($V * f$). On its next pass, it notices a relationship between this new (Vf) variable and temperature and thus creates ($V * f$) / T. On the next pass, no relations can be found and so the program is forced to remove the Vf/T description and backup to the previous level. Again no other relations can be found at this lower level and ABACUS backs up to its original starting level. No other relations can be found involving frequency and so other pairs are considered. This time ABACUS takes a step along the correct path by finding that velocity and temperature are proportionally related. Thus the new variable V/T is created and the program moves up once

again. As before, the program discovers the relation V//T and proceeds along this path. Once again it proves fruitless and ABACUS backs up. This time, however, we are backing up to the V/T level (not the V/ level as before) and another relation can be found, namely (V/T) * V. This, it should be noticed, is the relation we set out to find and ABACUS terminates after noticing that the new variable equals 401.322 for all data groups. In this manner, ABACUS was able to discover the relationship between the speed of sound and temperature, even when two other variables were present in the data.

```
a-events
  #      T          V          P          /
  1    330.000    363.918    0.750    3500.000
  2    330.000    363.918    1.000    9500.000
  3    330.000    363.918    2.000    10000.000
  4    320.000    358.362    0.800    6500.000
  5    250.000    316.750    0.800    1500.000
  6    250.000    316.750    1.000    1500.000
  7    250.000    316.750    0.750    1000.000
  8    250.000    316.750    0.750    1500.000
  9    273.000    331.000    1.000    1500.000
 10    273.000    331.000    1.000    3500.000
 11    273.000    331.000    0.750    3500.000
 12    273.000    331.000    1.250    2000.000
 13    273.000    331.000    2.000    2050.000
 14    273.000    331.000    0.500    2500.000
 15    273.000    331.000    3.000    3000.000

  a-outhypo
IF
    1
THEN
      V^2 = 401.322 * T
```

Fig. 3.

Notice that since the discovered relation held for all of the data given, ABACUS proposed that there are no constraints and that this law will always hold (indicated by a blank entry in the IF clause).

2.1.2. Overview

Data is presented to ABACUS in the form of tables. The first table provides some of the algorithm parameters. These include such things as the maximal search depth and the minimal percentage of the data a valid relation must hold for. The second table, "variables", merely lists all of the variable names and their type (numerical or nominal). Finally, the events" table is a row by row listing of observed events with the first row giving the variable names corresponding to the individual columns. No indication of relationships or independent/dependent constraints are provided.

Once the data is read in, AB (Fig. 4) is called repeatedly until all of the events have been empirically described or until no further relations may be found. Function AB modifies the search-space and global variables to help guide the search process. It first invokes the basic

```
FUNCTION AB (events) : boolean;

BEGIN
  finished := Findrelation(events);

  for group in nominal_subgroups(events) do
    if (not finished) then
      finished := Findrelation(group);

  if (not finished) then
      universal_ignore := nominal_variables;
      finished := Findrelation(events);
    end;

  if (not finished) then
      universal_ignore := nominal_variables + all_numeric_variables;
      finished := Findrelation(events);
    end;

  AB := finished;
END; (* AB *)
```

Fig. 4.

search routines using unmodified global information. If no relation could be found and nominal variables exist in the data, AB divides up the data events into nominal-subgroups and searches for relations on each individually. A nominal-subgroup is a set of events whose nominal variables all take on the same values. If no relation is found for any of the subgroups, AB effectively removes all nominal variables from the event data by including them in a "universal-ignore" set. Section 2.1.4 explains the details on why the presence of nominal variables may interfere with the search process. Finally, if no relation has been found, all variables are included in the universal-ignore set to remove any dependent/independent constraints imposed by the search algorithms.

Function Findrelation (Fig. 5) is the main search driver for the AB routines. For each permutation of variable pairs (col1, col2), Relation is invoked to determine the proportionality of the pair. Thus col1 may be directly proportional to col2, indirectly proportional to col2, or no proportionality can be discerned (indeterminant). If it is estimated that a relation exists, a new variable is created for this relation (col1*col2 or col1/col2) and the new variable is checked for terminating conditions (see Sec. 2.1.6 for a description of terminating conditions). If immediate cessation is indicated, the algorithm will stop. If the new variable's measure of constancy" is above a user-specified threshold, the algorithm will stop after exploring higher levels of relations. Otherwise, higher levels of relations are likewise explored with the exception that the permutation process continues if no relations are found.

So that we might speed up this exhaustive search process and attempt to achieve maximal advancement at each level, the permutation process is heuristically ordered to lead ABACUS along a "most probable" and "most promising" path first and along continually

```
FUNCTION Findrelation (events) : boolean;

VAR
    col1    : variable_column;
    col2    : variable_column;
    rel     : (direct, inverse, indeterminant);

BEGIN
  order_of_search[1] := non_related_variables;
  order_of_search[2] := given_variables - non_related_variables;
  order_of_search[3] := new_generated_variables;

  col2 := newest_variable;

  for search_set in order_of_search do
    repeat
      col1 := pop(searchset);
      rel := Relation(col1, col2);
      if Useful(col1, rel, col2)              (* no cancellations *)
        then begin
             Create(col1, rel, col2);         (* create new variable *)
             finished := Check_endconditions;
             stop_flag := Check_constancy;
             if (not finished) then           (* explore higher levels *)
               finished := Findrelation(events);
             if (not finished) then
               finished := stop_flag;         (* stop if good enough *)
           end;
    until finished or (search_set = []);

  Findrelation := finished;
END; (* Findrelation *)
```

Fig. 5.

decreasing levels of promise with each subsequent pair. To this end, the permutation space is divided up into 3 sections. The first section is a set composed of those original input variables which are still not part of any level of description. Thus, when discovering the ideal gas law (PV/nT = Constant) and at the PV/n level, ABACUS first examines the relationship between PV/n and T, saving the time that would be wasted if P, V, and n were examined first. If no relation is found after exhausting this first set, the search routine then takes variables from the second set which contains all of the other original input variables. It is assumed from these first two sets that using original input variables will lead to more promising paths than will using newly generated variables. If still no relation is found ABACUS will then go on to examine all of the previously generated variables. As was stated earlier, these generated variables may serve not only as descriptions for levels below them, but also as data for the levels above.

2.1.3. Nominal Subgrouping - Ability to Divide & Conquer

Often, when a relation is only present in a portion of the data and the data is examined as a whole, no apparent relation can be seen or perhaps a partial relation will be found which is the opposite of the correct solution. To account for these possibilities, if after examining the entire search space no terminating condition has been found, ABACUS will proceed to attempt to divide the data into groups containing at least two events, where the nominal variables for a group will all have the same value. This might be compared with the intrinsic property concept of the BACON systems in which the program takes into account the idea of an intrinsic property identifiable with a nominal variable value. ABACUS will then proceed to examine each of these groups individually as if they were the entire dataset by themselves. Relations which were hidden by the noise of the other groups may now be discovered.

As an example, let us look at the ideal gas law. This old law of Chemistry relates the temperature, pressure, volume, and number of moles of an ideal gas by the formula $PV = nRT$ where R is the universal gas constant ($R = 8.32$ newtons/moles kelvin). An additional variable *state,* which takes on the values *gas* and *solid,* will be added and, because the ideal gas law only applies to gases, imaginary values are selected for the *solid* cases in this example while the *gas* cases are made to adhere to the law. For this example, ABACUS proceeds as before, but this time an odd block of data has been added, creating noise and making it appear as if there were no relationship between any of the variables. The entire search space is examined and ABACUS is forced to give up without having discovered any relation. The program then proceeds to divide up the data into two subgroups according to whether the state variable is equal to *gas* or *solid.* Being alphabetically first, the *gas* subgroup is now examined. From this point on, all of ABACUS's routines see the data as only this subgroup and do not know of the existence of the *solid* group. Since the data currently under consideration adheres to the stated law, the PV / nT relationship of the ideal gas law is discovered. Next, the solid data is examined and some miscellaneous relationships are found to hold (the classes b, c, and d). While these miscellaneous relations are semantically uninteresting, they are correct and ABACUS treats them the same as the discovered gas law relation (Fig. 6).

---

a-events

#	STATE	V	N	T	P
1	gas	4992.000	1.200	330.000	0.660
2	gas	4992.000	1.200	350.000	0.700
3	gas	4992.000	1.200	280.000	0.560
4	gas	4992.000	1.200	250.000	0.500
5	gas	1664.000	1.000	300.000	1.500
6	gas	1664.000	2.000	270.000	2.700
7	gas	1664.000	2.000	350.000	3.500
8	gas	4992.000	0.800	375.000	0.500
9	gas	2496.000	1.000	300.000	1.000
10	gas	1664.000	2.000	250.000	2.500
11	gas	1664.000	2.000	310.000	3.100
12	gas	4992.000	1.000	300.000	0.500
13	gas	3328.000	1.000	300.000	0.750
14	gas	1996.800	1.000	300.000	1.250
15	gas	2496.000	1.132	320.000	1.208

```
b-events
  #    STATE        V          N          T          P
  1    solid        1690.000   20.000     100.000    2.000
  2    solid        1690.000   20.000     150.000    2.000
  3    solid        1690.000   20.000     160.000    2.000
  4    solid        1680.000   20.000     220.000    2.000
  5    solid        1670.000   20.000     210.000    2.000
  6    solid        1660.000   20.000     200.000    1.000
  7    solid        1660.000   20.000     200.000    2.000
  8    solid        1660.000   20.000     210.000    3.000
  9    solid        1660.000   20.000     220.000    3.200
 10    solid        1400.000   19.000     200.000    1.000

c-events
  #    STATE        V          N          T          P
  1    solid        4992.000   42.000     150.000    0.500
  2    solid        4992.000   42.000     160.000    0.600
  3    solid        4992.000   42.000     170.000    0.700

d-events
  #    STATE        V          N          T          P
  1    solid        1400.000   17.000     110.000    2.200
  2    solid        1200.000   16.000     110.000    2.200
```

```
a-outhypo
IF
    1    [STATE=gas]
THEN
     V * P = 8.320 * N * T

b-outhypo
IF
    1    [N=19.000..20.000]
THEN
     N^3 = 4.793 * V

c-outhypo
IF
    1    [N=42.000]
THEN
     N^3 = 14.841 * V

d-outhypo
IF
    1    [N=16.000..17.000]
THEN
     N^2 = 0.210 * V
```

Fig. 6.

2.1.4. Ignoring Nominal Data

During the discovery process, when two variables are examined for the possibility of a relation between them, all variables which are not part of the current level of description are held constant. For example, if the variable 'PV' is being compared to the variable 'n' in the ideal gas law, the variable 'T' must be held constant. Notice that neither 'P' nor 'V' can be held constant because they are subcomponents of the 'PV' variable. Because nominal

variables are never included in a level of description, they are held constant throughout the discovery process. Likewise, they are not involved in the backtracking process. They thus have the potential for getting in the way of the search mechanism. As an answer to this problem, ABACUS is able to remove all nominal variables from consideration and thus make them essentially invisible. If no terminating condition is found after searching the data as a whole and after searching the data using the nominal subgroup approach, another attempt is made to find a proper relation by starting over with all nominal variables removed.

As an example, let us observe the steps followed by ABACUS in discovering Kepler's law of planetary motion. Kepler's law states that a very simple relation exists between a planet's period of rotation around the sun (P) and its distance from the sun (D). This relation is formally stated as follows, where the value of the *Constant* varies according to the units used:

$$\frac{D^3}{P^2} = Constant$$

```
a-events
   #      PLANET        D          P
   1      mercury       0.387      0.241
   2      venus         0.723      0.615
   3      earth         1.000      1.000
   4      mars          1.524      1.881
   5      jupiter       5.203      11.862
   6      saturn        9.539      29.458
   7      uranus        19.191     84.015
   8      neptune       30.071     164.788
   9      pluto         39.597     249.170

  a-outhypo
IF
    1
THEN
     D^3 = P^2
```

Fig. 7.

A quick examination of the data (Fig. 7.) will reveal that there is no way for ABACUS to hold the value of the *planet* variable constant and at the same time note changing relationships between the other two variables. Because of this property in the data, the program quickly gives up when it proceeds to examine the data as a whole. Similarly, it must give up again when it attempts to divide the data up into nominal subgroups. This is due to the fact that each subgroup has only one member. As a next resort, ABACUS removes the *planet* variable from future consideration. With *planet* out of the way, Kepler's law is quickly found. In the above data, D (semi-major axis) is expressed in multiples of the earth's distance from the sun and P is expressed in years.

2.1.5. Removing Dependent/Independent Constraints

As stated earlier, holding certain variables constant while looking for relations between others can conceivably interfere with the discovery process. This is especially true for noisy

irrelevant variables which often don't appear in nice patterns. Likewise, it is often the case that not enough examples have been provided so that some variables have the same value while others are changing. Because of this possibility, as a last attempt before giving up, ABACUS removes all restrictions to force dependent/independent relations on the variables. Under this mode of operation, variable pairs are examined ignoring what values the other variables in the data possess. This creates less certainty in the search process and therefore is only attempted after all of the previous methods have failed.

2.1.6. Termination Conditions

Up until now, it has merely been stated that ABACUS quits once it reaches a terminating condition. It is time to examine what these conditions are and how they affect the search process.

There are three different terminating conditions. The most obvious and already mentioned condition is when a new variable is equal to a constant value for the entire set of data. At this point, searching is terminated immediately. There is also one other condition which will cause immediate cessation of the current AB run. If at any time a variable is found to be constant for a nominal subgroup, search will stop at this point and new event sets are created, removing these events from the data. This occurs whether the program is examining the entire set of data as a whole or is in the secondary process of examining the nominal subgroups separately.

Finally, the program will stop if a predefined percentage of the new variable's values are a constant. This percentage may be input as a parameter to the program, or the default value of 40% will be used. As was mentioned earlier, ABACUS is able to find a relation which does not necessarily hold for all of the data. With this ability comes the problem of knowing when good is good enough. Therefore a constancy percentage parameter is provided so that the user can specify what a good level of constancy should be. If the parameter is set at 100%, only a relation universally describing the data will be acceptable. At the same time, the chances for finding any relation at all are diminished. Note, however, that the nominal subgroup constancy condition stated above is tested first and ignores the constancy parameter. In an effort to find the highest possible percentage, this terminating concept is combined with the backtracking mechanism. Once a level of description is found which satisfies the given percentage, this level is marked as a point of no return and the program may not back down from this level. It does, however, continue to examine higher levels of description for better levels of constancy. During this time, if a level is discovered which has the same or higher degree of constancy, it is marked and represents the new point of no return. If no better description is found to exist along this path, the program will then return to this marked level and terminate with the specified level being the highest level of description.

In the ideal gas law example, ABACUS discovered the PV/n relation and found that it was equal to 2496 for a sufficient percentage of the data to be over the accepting threshold. Because it did not immediately stop here, the description PV/nT was found.

2.1.7. Tautologies - The Mathematical Cancellation Problem

Because the ABACUS search algorithm will exhaustively examine the data for a relation until a relation is found or until the entire search space has been explored, it was found early on in this project that a degenerate solution will always exist in a block of data if relations are

allowed to exist which, mathematically, are a step down in level of description from the level below. An example of such a degenerate solution occurred when ABACUS was first presented with the ideal gas law data which had the 'state' variable added. Because the correct relation could not be found when the data was examined as a whole, ABACUS cleverly came up with the formula

$$\frac{VTn^2}{VTn^2} = 1$$

which of course held for all of the data. Because relations will often be found that result in a numerator-denominator cancellation, a method for disallowing new descriptive levels which have this flaw had to be found. A crude algorithm was installed which prevented PV/n from being multiplied by 'n' and similar elementary cases. This algorithm, however, failed when ABACUS attempted more complicated tasks such as dividing PV/n by P/n. Finally an algorithm was written which was able to examine each individual variable and discover whether or not the proposed relation would cause a cancellation of this variable.

2.2. Aq Algorithm

The Aq algorithm was developed by R.S. Michalski and has been the basis of numerous research projects in machine learning here at the University of Illinois. It is a routine which takes as its input different sets of example events (class sets) and generates descriptions of these event sets which are called covers. Aq has been used successfully in expert systems to generate rules and is currently being used to guide a computer player of the card game Eleusis. It is a useful exercise in the study of automated machine learning.

There are several terms which must be defined in order to properly understand the Aq algorithm. The description which Aq generates for a particular event set is called a *cover*. This cover is a designation of the values certain variables possess for this event set which distinguish the set from the other event sets. A cover is composed of a single complex or a disjunction (logical OR) of complexes. A ***complex*** is a cover which describes a portion of the event space. For example, one complex might be given as [A=2]∧[B=3] which states that all events it describes are such that A always equals 2 and B always equals 3. Suppose for a particular event set 2 complexes are required to fully describe it. Then the cover might be given by ([A=2]∧[B=3]) ∨ [C=5]. Thus, all events in the event set under consideration satisfy at least one of the 2 conditions and no events in the other sets do satisfy the conditions. Finally, a complex, as shown, is a single selector or a conjunction (logical AND) of a series of selectors. A ***selector*** is a value or range of values that a single variable might possess. For example, the complex [A=2,3]∧[B=4] describes all events in which B equals 4 and A is either 2 or 3 and contains selector [A=2,3] and selector [B=4].

Aq will iteratively generate covers for each input set of events. When a particular event set is being covered, its members will be designated as the "Positive Events" and all of the other events are grouped into a "Negative Events" set. The primary routine is the "star" generating procedure which will generate a set of alternative complexes using a single (as yet uncovered) positive event as a seed. A particular problem might require that the final cover is a disjunction of these complexes. From the resulting star, the best complex is chosen based on some evaluation function. As a rule, a good complex is one which describes many positive events and contains only a few conjunctive selectors. Once the best complex is chosen from the star, this process is repeated until all positive events have been covered.

```
FUNCTION Cover  (lplus, lminus : eventset) : cover;

VAR
    seed    : event;
    star    : complexlist;
    best    : complex;

BEGIN
  while (lplus <> nil) do begin
    begin
      seed := head(lplus);
      star := Star (seed, lminus);
      best := Bestcomplex (star);
      result := append (best,result);
      lplus := Knockout (lplus,best);
    end;

  Cover := result;
END;  (* Cover *)
```

Fig. 8.

Function Cover will return a cover of the given event set, *lplus*, which discriminates this set from all of the other event sets that have been grouped into *lminus*. For each iteration, a single *seed* event is taken from the set of positive events (lplus) and a star is generated around this seed. The best complex is then chosen from this star and added disjunctively to the current incomplete cover. Finally, any positive events accidently covered by this latest complex are removed from lplus (Knockout) and this entire process repeats until there are no more positive events remaining to be covered. The lexicographic evaluation function (LEF) is used in Bestcomplex and elsewhere in the program to measure and make decisions on the quality of the given complexes. As stated earlier, the most desirable complex chosen from the star is generally one which covers the most positive events with the minimal number of selectors.

```
FUNCTION Star (seedevent, lminus : events) : complexlist;

VAR
   elemresult        : pcomplex;
   result            : complexlist;
   intsctsneg        : starbooleans;

BEGIN
  result := universe;

  for minusevent in lminus do
   begin
     if checkintrsctneg (result,lminus,intsctsneg) then
       begin
         elemresult := Generalize (seedevent, minusevent);
         result := Multiply (result,elemresult,seedevent,intsctsneg);
         result := Simplify (result);
       end;
   end;

  Star := result;
END; (* Star *)
```

Fig. 9.

Function Star generates a set of alternate complexes which discriminate a given seed event from a list of negative events. The idea behind the Star routine is to create maximal generalizations of the given seed event, while still remaining specific enough to never include a negative event in this description. Initially, the Star is set to describe the entire universe of the event space. Then, for each negative event, a generalized (extended) description of the seed is created. For example, if A and B are nominal variables, then the generalization of A over B is all possible values in the domain of A and B which are not equal to B. This generalized description would thus describe A as being all events in A's domain except those which are specifically in B. The new description is then added to the current star (Multiply) in such a way that the entire star is modified so that no complex covers the negative event currently under consideration. In this manner, the Star is made more specific with each iteration in a maximally general way. Finally, each time through the loop the star is simplified and trimmed down to a specified size by absorbing any redundant complexes and removing those which have been found to have low value. This trimming enforces a beam search methodology upon the generation process and prevents what would otherwise be a computational explosion.

3. ABACUS System

Up until this point, we have focused primarily on the independent workings of the two algorithms AB and Aq. We shall now explore the discoveries of the system as a whole and observe how these two processes work together. When ABACUS discovers a law which does not universally describe all of the input data with a single constant, the program separates the data into different classes, in the hope that Aq will uncover the characteristics of the data which caused these different classes to exist. Data 'tuples' obeying the same relation will thus be grouped together in the assumption that there is some trait they all have in common which the other groups lack.

3.1. Coulomb's Law

For our first example, we shall explore Coulomb's law of electrical force. When the charges of two objects have the same sign, these objects repel each other; if they have opposite signs, they attract each other. Electrical forces are among the most powerful in nature and yet we are rarely conscious of them in everyday life. This is due to the fact that the charges of the proton and electron are very nearly, if not exactly, the same while their signs are opposite. They thus tend to offset each other in nature. In a vacuum, Coulomb's law may be stated as:

$$F = \frac{1}{4\pi\epsilon_0} \frac{q_1 q_2}{r^2}$$

where $\epsilon_0 = 8.85 \times 10^{-12}\ C^2/Nm^2$ and is called the permittivity of free space. When the surrounding medium is not a vacuum, forces present in the material reduce this force and so ϵ must be defined to replace ϵ_0 and is a constant for a particular material. ϵ is called the permittivity of the material and Coulomb's law is restated as:

$$F = \frac{1}{4\pi\epsilon} \frac{q_1 q_2}{r^2}$$

For this experiment, ABACUS was given data in which the charge q_1, the charge q_2, the distance r, and the force F were related by Coulomb's law where ϵ equaled the permittivity of the material named in the 'substance' variable. Because of the size of a coulomb, q_1 and q_2 are given here in units of micro-coulombs (10^{-6}), while the force is in Newtons (N) and the distance r is given in meters. An irrelevant variable was introduced in the form of an imaginary mass value (m) which was intended to give the mass of the two charge carriers. Notice also (Figure 10) that an odd substance has been added to the data which possesses an entirely different relationship and has been added merely to complicate things.

a-events

#	SUBSTANCE	F	R	Q1	Q2	M
1	oddsubst	2.000	1.000	0.500	2.000	4.000
2	oddsubst	25.000	1.000	1.000	2.500	25.000
3	oddsubst	40.000	2.000	40.000	3.000	2.000
4	oddsubst	2.000	2.000	1.000	3.000	4.000
5	oddsubst	4.000	2.000	2.000	3.000	4.000
6	oddsubst	8.000	1.000	4.000	5.000	2.000
7	oddsubst	8.000	1.000	2.000	5.000	4.000
8	oddsubst	8.000	1.000	1.000	6.700	8.000
9	oddsubst	25.000	1.000	25.000	45.000	1.000

b-events

#	SUBSTANCE	F	R	Q1	Q2	M
1	water	7.025	2.000	500.000	500.000	44.000
2	water	2.810	1.000	50.000	500.000	56.000
3	water	1.405	1.000	25.000	500.000	67.000
4	water	2.000	1.500	100.090	400.000	85.000
5	water	20.000	2.000	50.000	14235.185	23.000
6	water	5.000	1.500	250.228	400.000	96.000
7	water	0.989	1.000	22.000	400.000	23.000
8	water	1.236	1.000	22.000	500.000	55.000

c-events

#	SUBSTANCE	F	R	Q1	Q2	M
1	air	35.555	2.000	500.000	31.652	4.000
2	air	20.000	1.500	50.000	100.151	3.000
3	air	35.555	2.000	719.369	22.000	3.000
4	air	20.000	2.000	50.000	178.047	2.000
5	air	2.000	1.500	22.762	22.000	1.000
6	air	2.000	1.500	22.000	22.762	5.000
7	air	35.555	1.500	50.000	178.047	6.000

d-events

#	SUBSTANCE	F	R	Q1	Q2	M
1	ice	2.000	4.450	22.000	840.766	45.000
2	ice	15.000	2.000	560.510	50.000	65.000
3	ice	20.000	2.000	747.347	50.000	44.000
4	ice	20.000	2.000	25.000	1494.694	87.000
5	ice	20.000	1.500	25.000	840.766	55.000
6	ice	2.000	2.000	22.000	169.852	45.000
7	ice	15.000	0.597	50.000	50.000	34.000

e-events

#	SUBSTANCE	F	R	Q1	Q2	M
1	silicon	1.696	2.000	22.000	404.644	21.000
2	silicon	1.696	2.000	50.000	178.047	29.000
3	silicon	2.000	1.000	22.000	119.300	22.000
4	silicon	20.000	1.500	50.000	1181.076	28.000
5	silicon	15.000	1.000	393.692	50.000	19.000
6	silicon	15.000	0.500	98.423	50.000	26.000
7	silicon	1.666	1.500	98.423	50.000	28.000

f-events

#	SUBSTANCE	F	R	Q1	Q2	M
1	germanium	20.000	0.176	22.000	50.000	20.000
2	germanium	1.405	2.000	25.000	400.000	18.000
3	germanium	1.405	1.000	50.000	50.000	15.000
4	germanium	20.000	1.500	200.182	400.000	12.000
5	germanium	1.405	2.000	22.000	454.555	15.000
6	germanium	15.000	1.500	150.137	400.000	13.000
7	germanium	0.702	1.000	25.000	50.000	16.000

ABACUS

```
  a-outhypo
IF
    1    [SUBSTANCE=oddsubst]
THEN
     M * Q1 = F * R

  b-outhypo
IF
    1    [SUBSTANCE=water]
THEN
     Q2 * Q1 = 8897.352 * F * R^2

  c-outhypo
IF
    1    [SUBSTANCE=air]
THEN
     Q2 * Q1 = 111.280 * F * R^2

  d-outhypo
IF
    1    [SUBSTANCE=ice]
THEN
     Q2 * Q1 = 467.160 * F * R^2

  e-outhypo
IF
    1    [SUBSTANCE=silicon]
THEN
     Q2 * Q1 = 1312.363 * F * R^2

  f-outhypo
IF
    1    [SUBSTANCE=germanium]
THEN
     Q2 * Q1 = 1779.015 * F * R^2
```

Fig. 10.

As can be seen in Fig. 10, ABACUS easily discovered the Coulomb relation and a totally different relation for the odd substance. It was also discovered that each of the five constants in the Coulomb relations corresponded to a different substance. With this observation, we may conclude that there is an intrinsic property associated with each substance such that Coulomb's law is equal to a constant which corresponds to that material. Hopefully, future work will enable the program to make and remember these types of simple conclusions. It should be pointed out that the constant term in each of the Coulomb relations is equal to $4\pi\epsilon$ for that substance.

3.2. Pendulum

This next example is useful in that its results are less obvious to a human observer and that some of the potential of the ABACUS system can be seen. The motion of a pendulum is described by the differential equation:

$$\frac{d^2\theta}{dt^2} = \frac{g\sin\theta}{L}$$

where θ is the angle the string makes from the perpendicular, L is the length of the string, and

g is the gravitational acceleration downward. If the angle of the pendulum's swing is small compared to the length L, it will very nearly undergo simple harmonic motion. For small angles, the sinθ term may be replaced by the close approximation θ and the equation may be solved for the pendulum's period T:

$$T = 2\pi \left(\frac{L}{g}\right)^{1/2}$$

Now let us present ABACUS with some data for different pendulums at various locations where the acceleration g has different values. L is given in meters while the angle θ is given in radians.

As can be seen (Fig. 11), ABACUS was able to discover the above relation. It should be noted that the constant 39.48 is merely the value of $(2\pi)^2$. This final relation, however, did not turn out to equal a universal constant for all of the data and no relation could be found for the remaining data.

When Aq examined the above classes, it discovered that the angle θ was the distinctive factor separating these two groups of data. Those data groups which did not hold the above stated relation all had much larger angles. We need only remind ourselves that the relation was intended as an approximation that was valid only when the angle θ was small - a distinction which the ABACUS system discovered.

a-events

#	L	g	T	theta
1	2.000	9.200	2.929	0.100
2	1.000	9.400	2.049	0.050
3	2.000	9.400	2.898	0.200
4	1.000	9.800	2.007	0.390
5	5.000	9.200	4.632	0.150
6	5.000	9.800	4.488	0.050
7	5.000	9.800	4.488	0.150
8	5.000	9.400	4.583	0.230
9	5.000	9.400	4.583	0.280
10	2.000	9.800	2.839	0.390
11	5.000	9.200	4.633	0.220

b-events

#	L	g	T	theta
1	5.000	9.800	4.650	0.700
2	2.000	9.400	3.010	0.700
3	2.000	9.200	3.090	0.850
4	2.000	9.800	3.000	0.800

a-outhypo

IF

1 [theta= 0.050.. 0.390]

THEN

T^2 * g = 39.479 * L

b-outhypo

IF

1 [theta= 0.700.. 0.850]

THEN

No formula holds

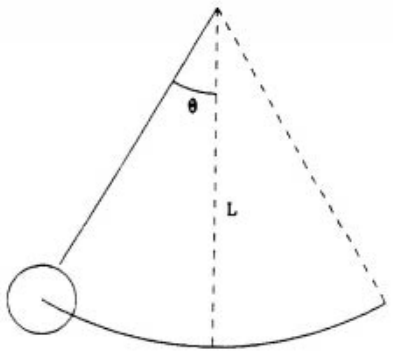

Fig. 11.

3.3. Ohm's Law

For the final example, we shall see how the system can be used to discover Ohm's Law and another intrinsic property. Ohm's law states that the current traveling through a wire is related to the voltage across the wire and to the resistance in the wire in the following way:

$$V = IR$$

In this experiment, ABACUS was given the values of the current I traveling through various wires of different resistance for a set of 3 different batteries named A, B, and C. Being a very simple relation, the formula I * R was discovered in one step and the resulting 3 class sets were passed to the Aq algorithm and the results in figure 12 were produced.

```
a-events
   #    BATTERY          I          R
   1    a              1.666      3.000
   2    a              2.500      2.000
   3    a              2.000      2.500
   4    a              5.000      1.000
   5    a              1.000      5.000

b-events
   #    BATTERY          I          R
   1    b              3.333      3.000
   2    b              4.000      2.500
   3    b             10.000      1.000
   4    b              5.000      2.000
   5    b              2.000      5.000

c-events
   #    BATTERY          I          R
   1    c              8.333      3.000
   2    c             12.500      2.000
   3    c             25.000      1.000
   4    c              5.000      5.000
   5    c             10.000      2.500

  a-outhypo
IF
    1    [BATTERY=a]
THEN
     R * I =  5.000

  b-outhypo
IF
    1    [BATTERY=b]
THEN
     R * I =  10.000

  c-outhypo
IF
    1    [BATTERY=c]
THEN
     R * I =  25.000
```

Fig. 12.

Notice that ABACUS has hypothesized that the final constant values are somehow dependent upon the battery being used. In fact, we can see that the constant value of 5 for battery A corresponds to its being a 5 volt battery. Similarly the voltage for battery B was 10 volts, while that for battery C was 25 volts.

4. Future & Limitations

The ABACUS system is a very useful exercise in simulating a part of the scientific discovery process. It has been designed to be as general as possible under the current implementation and has been shown to be applicable to a wide variety of chemical and physical laws.

This combination of two completely separate algorithms shows how different approaches to learning can be combined to create tasks neither could do alone. In experimentation, the pertinent variables are often mixed in with many completely irrelevant variables. With this design, a wide variety of variables may be introduced and decisions of pertinance may be left to the program. With Aq acting as a post-processor, the data can then be examined to answer questions concerning the applicable domain of the results and postulate the reasons for discovered differences in the data.

ABACUS will not always discover the intended laws hidden within the contrived data and in these cases will propose alternate relations to describe the data. After my initial frustration with this fact, I realized that there is still nothing invalid about the findings which the program did present. The results are still logically and numerically correct for the given data. My initial disappointment came when the program did not find the results I had intended for it to find. This, in the end, demonstrated that the system is indeed general. The algorithms, while working mostly on contrived data, do not demonstrate a contrived nature themselves and thus do not suffer from absolute predictability.

One definite failure of the current system occurred when a fellow student used the system to analyze data he was working on. He did not know if there were any empirical relations within his data and ABACUS, upon analyzing the data with various parameter changes, determined that indeed no empirical rules could be found within the data. The problem was further investigated, where it was discovered that when ABACUS is presented data representing a sparse matrix where a block within the matrix contains very simple empirical relations and the rest of the matrix (don't care area) is filled in with zeroes, ABACUS is both unable to find the relevant block within the sparse matrix and also unable to cope with the divide-by-zero" problem. In the future, I would like to investigate the possibility of an undefined" value for variables which will enable the algorithm to get past these problems.

The problem of noise was never properly addressed in this implementation. I do believe, however, that it might handle noise quite well. The relation discovering algorithms are extremely loose and would notice the existence of a relationship between two variables if the noise were not too severe. There are also two user specified parameters (dtrigger & itrigger) that affect the preciseness of this relation finding routine. In addition, the constancy determining routine defines a plus/minus interval to determine constancy and this interval may be altered as a parameter in the input as well.

The current system is limited to laws consisting solely of multiplication and division. That is, it is presently unable to discover laws which contain addition or subtraction. The obvious next step would be to modify the pattern finding routines to take into account slopes

and intercepts and thus additive relations. At this time, it is not known how involved such an undertaking might be, but I am optimistic. Since the algorithms employed, especially those which separate out class sets and determine class distinctions, are completely independent of what types of empirical rules may be found, I do not feel that this omission represents any compromise to the successes experienced so far. ABACUS was originally designed to solve the problems of irrelevant variables and relations which had specific domains within the given data. This task has been accomplished.

References

Becker,J.M. AQ-PROLOG: A Prolog Implementation of an Attribute-Based Inductive Learning System, Unpublished paper, University of Illinois, Dept. of Computer Science, December 1983.

Langely,P. Rediscovering Physics with BACON.3
Proceedings of the 6th International Joint Conference on Artificial Intelligence, pp. 505-507, 1979.

Langely,P., Bradshaw,G.L., Simon,H.A.
BACON:5 The Discovery of Conservation Laws
Proceedings of the 7th International Joint Conference on Artificial Intelligence, pp. 121-126, 1981.

Langely,P., Bradshaw,G.L., Simon,H.A.
Rediscovering Chemistry with the Bacon System.
In R.S.Michalski, J.G.Carbonell, and T.M.Mitchell (Eds.), *Machine Learning - An Artificial Intelligence Approach*, Tioga Pub., 1983.

Langely,P., Bradshaw,G.L., Simon,H.A., Zytkow,J.
The Search for Regularity: Four Aspects of Scientific Discovery.
To appear in *Machine Learning*, Volume 2,
R.Michalski, J.Carbonell, and T.Mitchell (Eds.), Tioga Pub., 1984.

Lenat,D.B. Automated Theory Formation in Mathematics.
Proceedings of the 5th International Joint Conference on Artificial Intelligence, pp. 833-842, 1977.

Lenat,D.B. The Role of Heuristics in Learning by Discovery: Three Case Studies. In R.S.Michalski, J.G.Carbonell, and T.M.Mitchell (Eds.), *Machine Learning - An Artificial Intelligence Approach*, Tioga Pub., 1983.

Michalski,R.S., & Larson,J.B. Selection of most representative training examples and incremental generation of VL1 hypotheses: The underlying methodology and the description of programs ESEL and AQ11, University of Illinois, UIUCDCS-R-78-867, May 1978.

Michalski,R.S. A Comparative Review of Selected Methods for Learning from Examples. In R.S.Michalski, J.G.Carbonell, and T.M.Mitchell (Eds.), *Machine Learning - An Artificial Intelligence Approach*, Tioga Pub., 1983.

Michalski,R.S. A Theory and Methodology of Inductive Learning.
In R.S.Michalski, J.G.Carbonell, and T.M.Mitchell (Eds.),
Machine Learning - An Artificial Intelligence Approach, Tioga Pub., 1983.

Winston,P.H. Artificial Intelligence, 2nd Edition. 1984 (p. 402-409).

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUC-DCS-F-84-1195	2.	3. Recipient's Accession No.
4. Title and Subtitle ABACUS Adding Domain Constraints to Quantitative Scientific Discovery			5. Report Date November 1984 6.
7. Author(s) Brian Falkenhainer			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Department of Computer Science University of Illinois 1304 West Springfield Avenue Urbana, IL 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No. NSF DCR 84-06801 ONR N00014-82-K-0186
12. Sponsoring Organization Name and Address			13. Type of Report & Period Covered 14.

15. Supplementary Notes

16. Abstracts

This paper describes the ABACUS quantitative scientific discovery system. ABACUS discovers mathematical relationships from a given set of numerical and nominal valued events. The system is able to subdivide these events into classes in terms of a mathematical formula which holds for a given class. ABACUS then generates descriminant descriptions of the classes, providing a domain constraint on each formula. The ABACUS algorithms are presented here (AB and Aq) and some of the results are shown.

17. Key Words and Document Analysis. 17a. Descriptors

Scientific Discovery, Quantitative Physics, Inductive Learning

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 28
	20. Security Class (This Page) UNCLASSIFIED	22. Price