Machine Learning 1: 367–401, 1986
© 1986 Kluwer Academic Publishers, Boston – Manufactured in The Netherlands

Integrating Quantitative and Qualitative Discovery: The ABACUS System

BRIAN C. FALKENHAINER (FALKEN @ A.CS.UIUC.EDU)
RYSZARD S. MICHALSKI (MICHALSK @ A.CS.UIUC.EDU)
Department of Computer Science, University of Illinois, 1304 West Springfield Avenue, Urbana, IL 61801, U.S.A.

(Received March 15, 1986)
(Revised April 17, 1986)

Key words: Quantitative discovery, inductive inference, learning by discovery

Abstract. Most research on inductive learning has been concerned with qualitative learning that induces conceptual, logic-style descriptions from the given facts. In contrast, quantitative learning deals with discovering numerical laws characterizing empirical data. This research attempts to integrate both types of learning by combining newly developed heuristics for formulating equations with the previously developed concept learning method embodied in the inductive learning program AQ11. The resulting system, ABACUS, formulates equations that bind subsets of observed data, and derives explicit, logic-style descriptions stating the applicability conditions for these equations. In addition, several new techniques for quantitative learning are introduced. *Units analysis* reduces the search space of equations by examining the compatibility of variables' units. *Proportionality graph search* addresses the problem of identifying relevant variables that should enter equations. *Suspension search* focusses the search space through heuristic evaluation. The capabilities of ABACUS are demonstrated by several examples from physics and chemistry.

1. Introduction

Research on inductive learning investigates the principles that govern the process of constructing knowledge from observed data. Significant progress has been achieved in developing methods for inducing symbolic, logic-style descriptions that characterize classes of examples or observations. Reviews of some such methods can be found in Dietterich and Michalski (1981) and Mitchell (1982). With the rapid expansion of expert systems applications, however, it is becoming clear that there is a need for utilizing in them not only qualitative knowledge, as has been the main thrust so far, but also quantitative numerical knowledge (Kowalik, 1986). This suggests that research in machine learning should develop methods for quantitative discovery, capable of automatically constructing numerical descriptions of the given phenomena. Such numerical knowledge would be a part of a deep model of the knowledge of an expert system.

There are also other reasons for research on quantitative discovery. For example, in many fields of science (especially in the life sciences), researchers gather empirical data as a prerequisite for building models and developing principles which explain the phenomena under study. Their tool box for analyzing the data has traditionally contained various statistical techniques, including regression analysis, numerical taxonomy, dimensional analysis, and the like. These methods manipulate numbers, equations, and similar structures without explicitly involving symbolic knowledge that represents domain constraints, control heuristics, underlying assumptions, etc. All this knowledge, if it ever enters the process, comes from the head of a data analyst. A given statistical procedure can only cope with specially prepared and interpreted numbers. Therefore, it seems very desirable to develop AI methods for data analysis which can reduce the amount of expert analysis currently required.

Some pioneering work in this direction has been done by Langley, Bradshaw, and Simon (1983a) with their BACON systems. Even earlier work has been done by Hajek and Havranek (1978) on the GUHA method of data analysis and by Zagoruiko (1976) and by Zagoruiko, Elkina, and Lbov (1985) on systems SPAR (1968) and PINCH (1978) for quantitative prediction of a variable and the simultaneous selection of the most informative attributes from the set of initial attributes.

This paper provides a comprehensive review of the issues in quantitative empirical learning and presents a methodology of such learning implemented in the program ABACUS. The ABACUS system is able to discover multiple mathematical equations for numeric data and derives explicit, logic-style descriptions stating preconditions for the application of the equations. Several new techniques particularly suited to quantitative learning are introduced in this work. *Units analysis* enables one to greatly reduce the size of the search space by examining the compatibility of variables' units. Two new search algorithms, *proportionality graph search* and *suspension search*, address some of the unique search problems associated with quantitative learning. Section 2 outlines the issues which arise in quantitative discovery. Section 3 discusses related work in the field, and Section 4 introduces the new approach taken in the ABACUS system. Sections 5 and 6 discuss the way in which ABACUS discovers equations and formulates preconditions for these equations. Several examples illustrating the performance of ABACUS are presented in Section 7. Finally, Section 8 overviews the ABACUS methodology and suggests directions for future research.

2. Goals for quantitative discovery

At the heart of quantitative discovery is the desire to induce mathematical descriptions that characterize the behavior of numerical observations. Independent of the technique used, there are a number of issues which any work in this area must address. Established disciplines, such as regression analysis, discuss ways for the data analyst to personally address many of these issues. Artificial intelligence techniques must attempt to automate this phase of the analysis. We therefore outline the following criteria for evaluating research on quantitative discovery.

1. *Coping with irrelevant variables.* In many discovery tasks, it is difficult to know which available variables are relevant to describing the observed events and which are not. A discovery program should be able to decide for itself what is relevant.
2. *Coping with incorrect and irrelevant observations.* In empirical data, it is often the case that some of the data is not representative of the process being observed. There are two common situations which give rise to this. First, some observations may simply be erroneous. Second, the process may not be defined outside a given range of values, such as the pressure being too high or the substance being in the wrong state.
3. *Discovering multiple equations underlying a collection of data and stating the conditions under which the equations apply.* It may often be the case that more than one equation is required to adequately describe a given set of observations. In these situations, the observations should be clustered around the various equations to form subsets of the original events. Conditions should be placed on each equation to describe when it is applicable.
4. *Handling different types of variables.* Often the observable variables in a given situation are both numeric and symbolic. When a discipline is young, for example, it may not be known that a given symbolic value has a one-to-one correspondence with an as yet undiscovered physical constant. A quantitative discovery system should take into account the symbolic information available.
5. *Imprecision and errors in the data.* The inaccuracy of experimental observations has always been a problem in science, and the discovery system should not be crippled because of it.
6. *Integrating with other learning systems.* While quantitative knowledge is quite valuable, it is only a part of the total knowledge available. These might include knowledge of qualitative dependencies among variables, including causal dependencies. A quantitative discovery system should be able to interact with discovery programs for acquiring different types of knowledge.
7. *Robust and efficient.* Discovery is inherently prone to combinatorial explosion. This, combined with the difficulties introduced by the criteria defined above, makes efficiency considerations particularly important for quantitative discovery. We want to be able to discover complicated equations and yet accomplish this in a reasonable amount of time.

3. Related work

Numerical data analysis and equation formation has traditionally used such standard techniques as regression analysis, numerical taxonomy, and dimensional analysis (e.g., Chatterjee & Price, 1977; Daniel & Wood, 1971; Huntley, 1952; Langhaar, 1951). These methods are very useful when the domain is well understood and when the observations correspond closely to the process being analyzed. For many situations, however, these methods are either difficult to use, requiring a great deal of human analysis, or they are simply inadequate. First, we are interested in discovering multiple equations in data when no single equation exists. Stepwise regression analysis is based on the implicit assumption that there is one best equation (Daniel & Wood, 1971). It will attempt to fit a single equation to the set of observed data, no matter how complicated the resulting equation. This leads to our second concern, comprehensibility. It is important that the results of a learning program be easy to understand. This is especially important in situations where a set of data may be described by two or more succinct equations or approximated by one complicated one. Third, regression analysis assumes that the data are a representative sample of the process being observed. Techniques are described in the standard texts to remove, by hand, nontypical data points, called *outliers* (Daniel & Wood, 1971; Chatterjee & Price, 1977). We are interested in the automatic removal of these data points. Fourth, we are interested in minimizing the amount of analysis required by the user. For the standard techniques, numbers and equations must be specially prepared and interpreted by the data analyst. Finally, we are interested in the smooth integration of quantitative and qualitative knowledge. Regression analysis uses *indicator variables* which take on values of 0 or 1 to represent different qualitative categories. We would like to see qualitative variables explicitly included in the discovery program's hypotheses.

The equation formation part of our research is related to the BACON project at CMU (Langley, 1979, 1981; Langley et al., 1981, 1983a; Langley, Zytkow, Simon, & Bradshaw, 1986), the COPER system (Kokar, 1981, 1986), and the HOTEP system (El-Shafei, 1986). The precondition formation part is related to the research on inductive learning done at the University of Illinois (Michalski and Larson, 1978; Michalski, 1983).

The BACON project began in 1978 with the construction of BACON.1; the most recent system is BACON.6. The basic approach taken in the early versions was to formulate empirical laws through the repeated application of hypothesis generation rules. Thus the ideal gas law ($PV/NT = 8.32$) would be formed in a layered fashion by creating the term PV, using this and the directly observed attribute T to form a more general term PV/T, and finally using this to formulate PV/NT which summarizes all of the given data. A variety of additions were subsequently made to this basic methodology. BACON.4 was able to postulate intrinsic properties for symbolic entities and could detect when common divisors existed for a variable's values. BACON.5 included a simple method for learning by symmetry. BACON.6 (Langley et al., 1983b, 1986) deviated from the methodology of the previous systems. The major difference was that forms of the law must be provided by the user, allowing the system to formulate more complex laws including trigonometric and algebraic functions.

While BACON compares well with the above criteria for quantitative discovery, several weaknesses can be pointed out. One limitation is that it cannot derive multiple equations to describe different subsets of the data. The issues of data clustering and the formulation of preconditions are not addressed. The data are assumed to be correct and relevant to a single process and the user must state which variables are dependent and which are independent. The equation formation techniques used in BACON.6 appear to be quite powerful, enabling the system to derive rather complicated laws in a straightforward manner. However, much of this power is achieved by requiring the user to provide a form of the answer.

A different approach to quantitative discovery is taken by Kokar (1981, 1986) and El-Shafei (1986). Central to this approach is the application of dimensional analysis (Langhaar, 1951; Huntley, 1952). Considering units of measurements, this analysis creates the set of all possible dimensionless products of variables provided in the data. These products are then used to form equations explaining the data. Traditional dimensional analysis requires that the relevant variables are known prior to application of the procedure, thus requiring extensive domain knowledge. Kokar solves the irrelevant variables problem by first trying to determine the completeness of the set of variables characterizing the given physical process. This step precedes the equation formation step and is able to discard irrelevant variables as well as detect when a needed relevant variable is missing. El-Shafei effectively ignores the need for determining variable relevancy prior to dimensional analysis by using regression analysis to form the desired equation from the set of dimensionless terms. He assumes that terms involving irrelevant variables will automatically drop out during the regression analysis. This may be an oversimplification of the problem since great care is given to variable selection using classic regression analysis techniques (Daniel & Wood, 1971; Chatterjee & Price, 1977).

These systems are quite robust in that they will find an equation to fit the data, no matter how complex that equation may be. Therein lies one of the problems with these techniques. Because only a single equation is always fit to the data, these systems cannot detect cases where the data could be better described by two or more equations. In addition, these systems fail to take into account the symbolic information available as well as having problems with situations requiring dimensional constants. In the following sections, we describe an approach to quantitative discovery which contains aspects of BACON, dimensional analysis, and symbolic approaches to inductive learning.

4. The ABACUS approach to quantitative discovery

There are many strategies to derive an equation or set of equations summarizing the behavior of some physical process. In choosing a particular strategy, one must weigh the gains from the use of that strategy against the losses. The approach taken in ABACUS has been to satisfy as many criteria from our list for quantitative discovery as possible, and to reduce the user supplied information to a minimum. ABACUS can handle irrelevant variables, symbolic variables of different types, and a certain degree of noise. Its great advantage is that it is able to discover multiple equations and ignore irrelevant observations. The only information required from the user besides the actual observations is a list of the attributes, their type (numeric or symbolic), and optionally their units (e.g., meters/second). The program is never told which variables to treat as dependent and which to treat as independent. In achieving these abilities, some sacrifices have had to be made in robustness and efficiency. The experimental results described in Section 7 indicate that the system is both general and powerful.

The ABACUS method of quantitative discovery consists of two steps. First, the *equation discovery module* analyzes the original empirical data and attempts to derive equations summarizing the observed behavior. If more than one equation is required to describe the observations, the data are divided into disjoint subsets, and equations are determined for each subset. The second step passes the resulting subsets to the *precondition generation module*. This module derives a logic-style description for each subset. Such a description is used as a precondition for each equation. The result is a series of if–then rules in which the 'if part' states the precondition for applying the rule specified in the 'then part'.

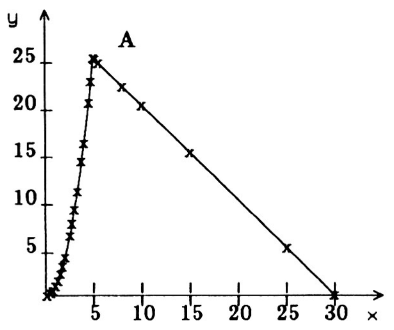
The equation learning module searches for the best equation to describe the given empirical data.[1] If the discovered equation holds for all events, the learning task is completed, and no preconditions need to be generated. If the equation describes only a subset of the events, however, then the subset described is removed from the list of events and associated with the equation describing it. Sometimes several classes of events can be described by one expression that evaluates to different values. When this occurs, a number of classes are formed, one for each value of the expression. Remaining events are passed to the equation learning procedure again in order to determine a separate equation for them. This iterative process repeats until all events are accounted for. When no equation can be determined for some events, they are placed in a 'miscellaneous' class.

Once the data have been divided into classes, the precondition generation algorithm is used to create discriminant descriptions for these classes. The resulting logical expressions can be used to predict which equation should apply to a newly observed event. The following example is used to illustrate the general algorithm used in ABACUS.

Suppose the system is given the data depicted in Figure 1(a). Observed values for x and y are read in and the equation discovery module is invoked. As there are only

[1] There are many ways to determine the *best* equation. Here we refer to the equation describing the largest subset of the data (i.e., the most general). It is also important to consider syntactic and domain-dependent criteria, such as the equation's simplicity or its relation to known physical phenomena, but the current work has not yet addressed these methods.

(a)



(b)

Rule A	*IF*	[x = 0.10 .. 5.00]
	THEN	$x^2 = y$
Rule B	*IF*	[x = 5.10 .. 30.00]
	THEN	x + y = 30.00

Figure 1. ABACUS analysis of graph example.

two variables, the space of possible equations is small. The best equation found, which describes 70% of the data is, is $x^2 = y$ (a discussion of the equation formation technique is in Section 5). Events covered by the equation are put in a class associated with this equation. The equation discovery module is invoked again to analyze the remaining events. This time, $x + y = 30$ is found to hold for all events and a class set is created for these events. Because all observations are accounted for, the equation discovery step is completed and the precondition module is called. This module searches for properties of the data which distinguish between the two classes. The results are presented in Figure 1(b). They state that when x is below 5, the equation is $y = x^2$, and when x is between 5 and 30, the equation $x + y = 30$ holds.

5. Discovering equations

The technique used in ABACUS depicts quantitative discovery as a search through the space of equations that could possibly describe the behavior of the observed data. This search process mathematically combines variables representing terms to form new terms. For example, x and y might be combined to form $x + y$. Search in this domain is different than in many other domains because new nodes are formed by the combination of existing nodes rather than by node expansion. In addition, we have not been able to derive a good heuristic evaluation function which can accurately indicate the possibility that a given node lies on a promising path. Before describing the search algorithms used in ABACUS, we first discuss how nodes in the search tree are formed, how search through a potentially exponential search space is constrained, and how the goal node is recognized once it is found.

5.1 Variable dependencies and proportionality graphs

At the heart of quantitative discovery is the concept that one variable's values may be dependent in some way upon the values of another variable. The early BACON systems looked for monotonic relationships in the data to create new hypotheses (Langley, 1981; Langley et al, 1983a). Michalski (1983) defines the M-descriptor stating that, if two variables exhibit a monotonic relationship, one should investigate the properties of their product. In the strict sense, variable x monotonically increases with y if the values of x always rise when the values of y rise while holding all other variables constant. There are two problems with such a strict definition. First, for a given set of data, it is not always possible to observe changing values of x and y while holding all other variables constant. Second, we must allow for inaccuracies and errors in experimental data. As a result, we are interested in the degree with which x is proportional to y rather than detecting if x exhibits a monotonic relationship to y for all of the data. With this in mind, we say that x is *qualitatively proportional* to y if, for a given percentage of the events (user specifiable), the values of x rise when the values of y rise while certain specified variables are held constant. Similarly, x and y are *inversely qualitatively proportional* if x decreases as y rises for a majority of the events under the same conditions. There are then four assertions possible as the result of a qualitative proportionality measurement:

$Prop^{+}$ (x, y)	— x and y are qualitatively proportional to a user-specifiable degree
$Prop^{-}$ (x, y)	— x and y are inversely qualitatively proportional to a user-specifiable degree
$Prop^{?}$ (x, y)	— insufficient data to determine if x and y are related
Norel (x, y)	— x and y are not related

To make a qualitative proportionality assertion about variables x and y, ABACUS looks for general trends in the data. Since it is not always possible to hold all other variables constant, an *exclusion set* is defined to be the set of attributes which do not need to be held constant and is constructed by the program and the user. The user must recognize which variables simply cannot or should not be held constant. Similarly, when measuring the proportionality between variables x and y, the program recognizes that, if x is a program generated variable composed of user defined variables v and w, then v and w should be removed from the set of variables which must be held constant. Since they are necessarily dependent upon x, it would be impossible to hold v and w constant while changing x. The trend detection algorithm determines whether y rises or decreases as x rises when all user defined variables not in the exclusion set are held constant. It never tries to hold program generated variables constant. If no groups can be found where all of these variables remain constant, then *Prop*$^?$ must be asserted. For each of the groups found, a measurement is made of the monotonic relationship between x and y from which an average is obtained and used as the degree of proportionality between x and y. This measure is then used to assert *Prop*$^+$, *Prop*$^-$, or *Norel*. The proportionality criterion has a margin of tolerance, allowing a moderate degree of noise and a limited amount of *conflicting proportionalities*. Conflicting proportionalities occur when some of the data indicates *Prop*$^+$ (x, y) and some indicates *Prop*$^-$ (x, y). In Figure 1(a), there were 16 points given for the curve (*Prop*$^+$ (x, y)) and 7 given for the line (*Prop*$^-$ (x, y)), causing the program to initially assert *Prop*$^+$ (x, y). We have developed an algorithm that can handle the conflicting proportionalities problem in a more general manner, based on determining *breakpoints* in the monotonic relationship between variables. For the data in Figure 1(a), it would first determine the breakpoint A, and then process points to the left and right of A independently.

From these proportionality assertions we may construct an undirected graph, called a *proportionality graph*, where the nodes represent variables, and edges indicate the presence of a qualitative proportionality relation between their incident vertices (Figure 2). For our purposes, edges shall only be constructed for *Prop*$^+$ and *Prop*$^-$ relationships, and *Prop*$^?$ will effectively be treated as *Norel*. In Figure 2, a is proportional (+ or −) to b, but not proportional to c.

As explained in Section 5.5, we are interested in nodes which form cycles in such a graph. In this context, the term cycle refers to any biconnected components (Aho, Hopcroft, & Ullman, 1974) which may exist. A biconnected component refers only to the maximal cycles in the graph or, in other words, only those cycles which are not a subset of some other cycle. In Figure 2, the single maximal cycle (or biconnected component) consists of the set of nodes $\{a\ b\ e\ f\}$.

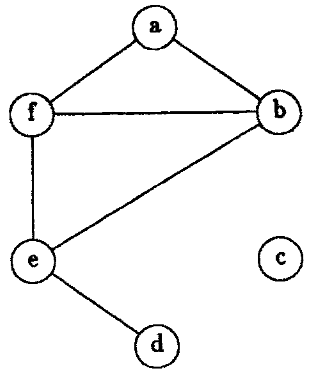


Figure 2. Proportionality graph.

5.2 Equation formation — a search for constancy

The existence of qualitative proportionalities between variables suggests the possibility of causal or other relationships between them. For example, if we knew that the value of x always goes down when the value of y goes up, then the relation $xy = constant$ might be binding these variables. This may be generalized to a rule:

If $Prop^{-}(x, y)$ then create a variable equal to xy

Such a variable is more likely to take on a constant value than x or y independently. Expanding on this concept, the following heuristics are formulated:

```
If $Prop^{+}(x, y)$ then
    Generate a variable equal to a quotient relation between $x$ and $y$      (5.1)
    Generate variables equal to difference relations between $x$ and $y$
If $Prop^{-}(x, y)$ then
    Generate a variable equal to a product relation between $x$ and $y$      (5.2)
    Generate variable equal to sum relations between $x$ and $y$
```

With these heuristics in mind, search in quantitative discovery involves the continual combination of variables which are qualitatively proportional to form new variables in the hope of finding a variable which takes on a constant value. Notice that application of the above heuristics tends to create variables with the same or higher degrees of constancy than the original variables.

The variables created from product and quotient relations are what one would expect. A variety of sum and difference relations may be formed, however, including $x + y$, $x^2 + y^2$, and $x^n + y^m$. Those actually generated will depend upon the units involved as well as other domain-independent constraints discussed below. Using these rules, the system can generate many new variables when qualitative relationships are detected among the current variables. In addition, ABACUS provides a facility for the user to predefine arbitrary functions or transformations on input variables which operate before the equation discovery module is called. For example, the user may instruct the program to replace all values of x with $\log(x)$ or to create a new input variable whose values are determined from a supplied function of existing variables.

5.3 Domain-independent constraints

Several domain-independent constraints are used to limit the large search space associated with quantitative learning. These constraints involve eliminating mathematically redundant expressions and physically impossible relationships. The constraints are divided into three categories:

- units compatibility rule
- redundancy detection
- tautology detection

Below we discuss each of these in turn.

5.3.1 Units compatibility rule

When the system decides to create new variables by firing the rules presented in the previous section, the additive relation rules will attempt to create a variety of new variables. Were all of these variables created every time one of the rules fired, the number of variables would explode and the search space would become unmanageably large. However, a simple physical constraint drastically limits the possible choices. For two entities to be added or subtracted they must be of the same type, that is, they should have the same physical units. One may divide meters by seconds, but one may not subtract seconds from meters. Therefore, any action that violates this *units compatibility rule* is blocked. This is similar in intent to the dimensional cohesiveness requirement of dimensional analysis (Langhaar, 1951; Huntley, 1952). All equations generated by ABACUS are guaranteed to be dimensionally cohesive if units are specified for each variable.

When attempting to generate sum relations between two variables x and y, if the units of x and y are equal, then terms such as $x + y$ and $x^2 + y^2$ will be created. If, on the other hand, the units of x and y are not the same, but differ only in exponent such that the units of x^n is compatible with the units of y^m, then the term $x^n + y^m$ (where $n \neq m$) would be created and terms like $x + y$ would be blocked.[2] Finally, if the units are not equal and cannot be made compatible by exponentiation, then no sum relations will be created, effectively blocking all instantiations of the sum generation rule. In practice, this is the usual case. It should be pointed out that these constraints only test the identity of units and provide no semantic interpretation to guide the search. In the future we would like to use here also constraints stemming from the physical properties of entities involved, such as trying to add the velocity of two unrelated entities.

5.3.2 Formula redundancy

A common side effect of combining existing variables to form new ones is the possibility that, for any new variable, a mathematically equivalent yet syntactically different expression defining a variable may have already been created. This is especially likely since variables created at one level in the search may be combined with existing variables from any other level. For example, say variable x represents the relation:

[2] A further constraint requires that the exponents n and m be less than 4. This is a heuristic limitation, but seems reasonable given that higher powers are rare in the natural sciences.

$$x = \frac{(ab)}{(cd)}$$

where the parentheses show that x was created by dividing a variable ab by another cd. Further, suppose during the course of the search the variable $b/(cd)$ had been created. At some point, the system will then try to create a new variable y:

$$y = a\left(\frac{b}{(cd)}\right)$$

As we can see, x and y represent the same variable, so creating y is redundant. From a purely syntactic examination, however, x and y are not equal. The solution to this problem is to use a canonical form for expressions so that equivalent formulas will always be syntactically equal. The form we use expresses all equations as a sum-of-products (Falkenhainer, 1985b). Thus $\frac{x}{y}(a-b)$ would be expressed as $\frac{ax}{y} - \frac{bx}{y}$. A canonical sum-of-products representation has also been used in the BACON systems to detect redundancy (Langley, 1981).

5.3.3 Numerical tautologies

Another problem with combining mathematical formulas is the possibility that a mathematical cancellation may result, causing the program to effectively take a step backward. Suppose, for example, the program discovers $Prop^{-}$ $(a/b, bc)$. Creating a new variable $\frac{a}{b}bc$ would result in b canceling out. Were such operations allowed to go unchecked, the system may soon discover that $\frac{abc}{abc}$ always equals 1 for any data given.

ABACUS allows no action which would result in a mathematical cancellation. Using the canonical form for formulas mentioned in the previous section, a check for tautologies is reducible to a set of simple logical conditions. If the tautology condition for a given operation holds, the proposed action is blocked.

5.4 Recognizing the goal

Because a valid equation may describe only a subset of the events, recognizing when a good equation has been found and when to terminate search is not as easy as it would be otherwise. There are three types of goal nodes recognized by the system. The first type corresponds to a term that describes all events, i.e., one which evaluates to the same value for every event (within a percentage range of uncertainty modifiable by the user). Such a goal is easily recognized and search terminates when one is discovered.

The second type of goal node is based on the notion of a nominal (symbolic) subgroup of events and also causes immediate cessation of the search process. A *nominal subgroup* is defined to be a set of events that are equal on all nominal

Table 1. Sample goal node recognition.

(a)

object	x	y	x/y
circle	2	2	1
circle	4	3	1.3
circle	6	4	1.5
box	2	1	2
box	4	2	2
box	5	2.5	2
box	6	3	2
triangle	3	2	1.5
triangle	5	3	1.6

100% Constancy in a Nominal Subgroup

(b)

object	x	y	x/y
circle	2	2	1
circle	4	2	2
circle	6	3	2
box	2	2	1
box	4	2	2
box	5	2.5	2
box	6	3	2
triangle	3	2	1.5
triangle	5	2.5	2

67% Constancy for Entire Event Space

attributes. If a term is found which evaluates to a single value for a nominal subgroup, search terminates on the assumption that an equation of significance has been found. For example, in Table 1(a), x/y has the same value for all events in the nominal subgroup corresponding to the object 'box'.

The third type of goal node does not halt the search algorithm. As each new variable is created, its *degree of constancy* is measured, and the variable having the largest degree of constancy is stored. The degree of constancy is defined to be the percentage of the data for which the function evaluates to a single value within a percentage range of uncertainty modifiable by the user. In Table 1(b), x/y has a 67% constancy because six out of the nine events are equal to 2. If two variables have the same constancy value, only the first discovered is remembered as it is more likely to be of a simpler, and thus more desirable, form. A more thorough approach would examine the equations according to various syntactic criteria and keep those that are both general and appealing. If search exceeds the allowed limit, the term having the highest degree of constancy is returned. If its constancy is greater than a user modifiable threshold, the resulting equation is reported. Otherwise, the program states that no formula could be found.

5.5 Search

ABACUS discovers equations by searching through the space of possible terms which relate the user supplied variables. These terms are formed by applying the variable generation rules to the current set of proportionality assertions. Even with domain-independent constraints, a search space generated in this manner can become quite large. In an effort to counter this problem, ABACUS uses a combination of two search algorithms which have been designed with quantitative learning in mind. The first algorithm, *proportionality graph search*, uses the graphical nature of the proportionality assertions to guide the search path and discriminate against irrelevant variables. The second algorithm, *suspension search*, enables the program to reduce the number of terms being examined by removing those that do not look promising until all other possibilities have been exhausted.

In this section we will examine only the search process itself, ignoring operations done once a final term has been selected. For illustration, two examples will be used. The first example deals with discovering the ideal gas law:

$$\frac{PV}{nT} = 8.32$$

where P is the pressure of the gas, V is the volume, T is the temperature in degrees Kelvin, and n is the number of moles. The ideal gas law equation belongs to the class of relations consisting solely of multiplication and division, and whose variables are all of degree one. It is a law which has been discovered in a variety of ways by the BACON programs (Langley, 1981; Langley et al., 1986).

The other example is the nonvector form of the conservation of momentum law:

$$m_1v_1 + m_2v_2 = m_1'v_1' + m_2'v_2'$$

This relation represents those equations which include addition and subtraction. It states that, when two particles collide while traveling along the same line, their total momentum is the same before and after the collision. To complicate the example, the masses m_1 and m_2 will be allowed to change after impact producing m_1' and m_2'. When the masses do not change, reducing the number of variables to 6, ABACUS discovers the equation in much less time.

5.5.1 Proportionality graph search

Experience has shown that, in terms of difficulty, the types of equations ABACUS is able to discover may be divided into two categories. Equations composed solely of multiplication and division tend to be easiest to discover, while those including other operators, such as addition and subtraction, tend to be more difficult (Falkenhainer, 1984, 1985b). Proportionality graph search is designed to handle equations falling in the first category. They correspond to a large percentage of the physical laws found in elementary physics and chemistry texts. Proportionality graph search is based on the observation that these equations will form a cycle in the corresponding proportionality graph, barring the presence of an exorbitant number of *Prop*$^?$ assertions. As an example, Equation 5.3 below represents a general equation of this type.

$$\frac{uvw^2}{xyz} = constant \tag{5.3}$$

Holding the four variables u, v, w, and x in Equation 5.3 constant and varying y will necessarily cause z to vary as well, in a direction which is completely predictable given the direction of change of y. This is true for Equation 5.3 no matter which four variables are held constant and which two are allowed to vary. In the absence of *Prop*$^?$ assertions, each variable is therefore qualitatively proportional ($+$ or $-$) to the other five. For the given problem, that may have more than six variables, the subgraph for vertices (u, v, w, x, y, z) must therefore be strongly connected and these nodes will thus form a cycle. This introduces another observation about the proportionality graph for such an equation. Irrelevant variables are more likely to be excluded from the above cycle and may often be incident on only one edge.

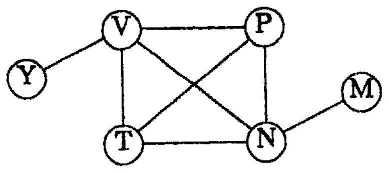
The proportionality graph search technique directs its search to the interrelations of variables forming a cycle and avoids variables that are not contained in a cycle. The algorithm consists of the repeated application of the following steps:

1. Form a proportionality graph for the current set of variables, both those provided by the user and those generated by the program. Exclude all edges which occurred in previously generated graphs.
2. Extract the cycles (biconnected components) and represent each cycle by the set of nodes it contains.
3. Search each cycle in a depth-first manner for a depth given by the cardinality of the set.

This process repeats until a suitable relation is found up to a maximum of K times.[3] For each graph, the cycle sets are sorted in decreasing order under the assump-

[3] The default search depth, K, is 4 since powers greater than 4 are seldom seen in the natural sciences.

(a)



(b)

Cycle Sets:

{ (P V) (P N) (P T) (V N) (V T) (N T) }

{ (Y V) }

{ (N M) }

Figure 3. Proportionality graph for ideal gas law ($PV/NT = 8.32$).

tion that the largest cycles will prove to be the most promising. A cycle (e.g., $\{V, N, P, T\}$) is searched in a depth-first manner by first removing two nodes that are proportional and combining them according to the equation formation heuristics to form new terms (e.g., $\frac{V}{N}$). The remaining nodes (e.g., $\{P, T\}$) are then tested one at a time against these terms to form new terms. For the set $\{P, T\}$ and the current node $\frac{V}{N}$, P would be tested against $\frac{V}{N}$ to possibly create new terms such as $\frac{PV}{N}$. If backtracking occurred, then T would be tested against $\frac{V}{N}$. This process repeats until either a solution has been found or until all combinations have been exhausted. Because nodes are removed from the cycle set as search progresses, powers of variables are not possible after the first round of search.

As an example of the heuristic power of this search technique, a sample proportionality graph is shown in Figure 3(a) for the ideal gas law, where a total of six attributes were initially provided by the user. As can be seen, the irrelevant variable mass, M, is independent of pressure, volume, and temperature, but is proportional to the number of moles of gas present. A similar situation exists for the variable Y. The three cycles of the graph are given in Figure 3(b), where solitary edges are simply treated as 'cycles' having only one edge. Figure 4 shows the search tree resulting from the above strategy applied to this example. The nodes shown were the only ones examined by ABACUS. For the ideal gas law, the program generated the minimum number of nodes possible to arrive at the correct solution.

While the proportionality graph search is quite adept at locating relations like the ideal gas law, this example happens to be ideally suited to such a search technique. Other types of relations, even those composed solely of multiplication and division but with higher powers, are not so well suited to proportionality graph search. For each iteration of the search algorithm, a new proportionality graph is constructed. The difficulty begins with the second graph constructed, and becomes increasingly

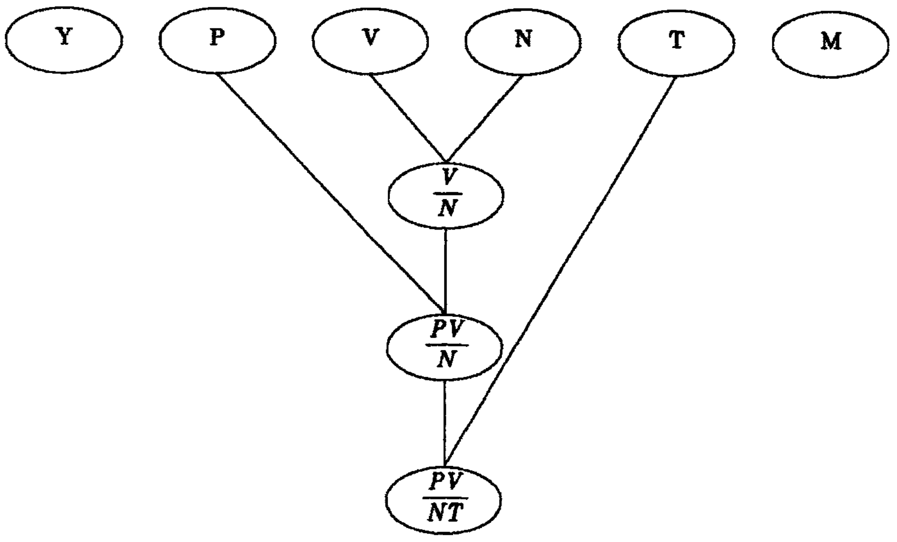


Figure 4. Proportionality graph search path for ideal gas law example.

worse with successive graphs. After completion of the first search pass, a large number of terms may exist in the system, many of which differ only slightly. Consequently, the second proportionality graph constructed has a much larger number of nodes than the first and, due to the similarity of the nodes, the graph tends to be highly interconnected. Therefore, the extracted cycles are quite large, sometimes encompassing the entire graph. As the depth of each cycle search is given by the number of nodes in the cycle and backtracking must be allowed, a great deal of time is wasted exploring very deep levels of the search tree.

5.5.2 Adding suspension search

To avoid the problems caused by repeated applications of proportionality graph search, ABACUS uses only one iteration of the algorithm. If no law is found, then the program employs a technique called suspension search. This algorithm is able to remove nodes from consideration, yet allows their return should they be needed. It combines the benefits of a beam approach with the allowances for faulty heuristics provided by backtracking. Suspension search begins as a normal breadth-first search. At each level, however, the values for each node are examined. As the ultimate goal is to find a variable whose values are constant or nearly constant, nodes possessing some degree of constancy are more likely to lie on a terminating path than nodes which lack any degree of constancy. To this end, when each level is created, all nodes on that level are divided into *active nodes* and *suspended nodes*. Suspended nodes are those whose constancy is less than a low threshold which is roughly 5% of the events.

Search then proceeds on to the next level, where only the active nodes of previous levels are visible to the search algorithm. The next level is created by testing the proportionality all new active nodes of the current level have among themselves and with all active nodes of earlier levels (as well as with old active nodes of the current level, as will be seen shortly). If no relation has been found by the time the depth limit (user specifiable) is reached, the best relation found so far is returned if its level of constancy is high enough according to a user supplied parameter. If not, search backtracks to the previous level where its suspended nodes are now activated and related to each other, its old set of active nodes, and those active nodes of earlier levels. Search then returns to the next level with a new set of active nodes. If still no relation is found, backtracking will go back farther and the process will repeat as before. An environment of each level is maintained to enable the program to remember what nodes were previously active and suspended when search returns. The suspension search algorithm is presented in Table 2. When invoked initially, nodes created during proportionality graph search join the user defined variables in level 1 to form the initial set of active and suspended nodes.

Because suspended nodes are ignored, fewer nodes are involved in the search at any one time. Therefore, search may be allowed to explore deeper than it could otherwise. A second search depth limit is defined, called the *filter depth*, which cites a limit shallower than that of the absolute depth limit. Search may proceed beyond the filter limit depth, but only active nodes are allowed for levels beyond this limit. Suspended nodes created at these levels are permanently discarded.

A partial suspension search tree is given in Figure 5 for the example involving the discovery of the law of conservation of momentum. The dashed horizontal line represents the filter limit depth which has been set to 3. A number of nodes may be eliminated as a result of this technique, considerably reducing the search cost.

Table 2. Suspension search algorithm.

```
FUNCTION Suspension (active_ancestor_nodes, active_nodes, suspended_nodes, environment)

•  If the search depth limit has been reached
     then return true if the best constancy found is greater than a threshold else return false
•  If new active or suspended nodes can be created from the current list of active nodes
     then return true if one of these has a constancy of 100%
     or return true if a call to Suspension using the new nodes returns true
•  If the filter depth has been reached
     then save the environment and return false
•  If new active or suspended nodes and be created from the current list of suspended nodes
     then return true if one of these has a constancy of 100%
     otherwise save the environment
         and return true if a call to Suspension using the new nodes returns true
•  Save the environment and return false
```

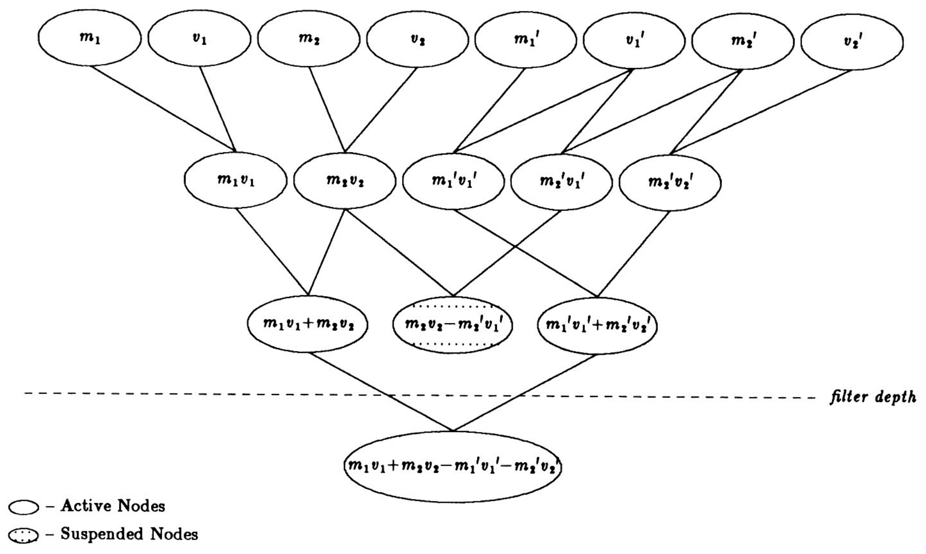


Figure 5. Partial suspension search tree for conservation of momentum.

Combining the proportionality graph search algorithm with the suspension search algorithm favors quick discovery of laws which are composed solely of multiplication and division while still being adept at discovering more complicated equations in a reasonable amount of time. As cycles in the first pass can never be larger than the number of given attributes, the depth-first search of the first phase is not deep for most problems, thus creating variables which would normally be created for more complicated examples anyway.

5.5.3 Analysis

Search algorithms representing all the possible combinations of proportionality graph search, breadth-first search, and suspension search have been constructed and directly tested in the ABACUS system. Empirical evidence has shown that, among these, the most powerful strategy is the combination of algorithms presented above (Falkenhainer, 1985b). Search strategies not including the proportionality graph search algorithm tended to be slower for examples such as the ideal gas law while the algorithm by itself was slower on most other examples. Suspension search proved to be equivalent to breadth-first search on small examples and superior on large examples.

The method shown is quite flexible. Where it falls short is in robustness. The only equations that can be discovered are those composed of variables raised to some integer power and combined through the use of multiplication, division, addition, and subtraction. While this is usually sufficient for introductory treatments of the physical sciences, we would like to strengthen this aspect of the current implementation in future research.

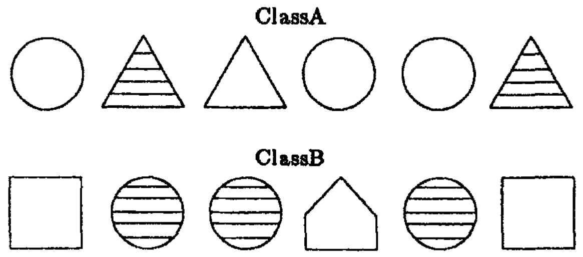
6. Formulation of qualitative preconditions

When multiple equations are discovered for a given set of data, ABACUS generates a logical precondition for each equation which describes when the formula is applicable. Deriving preconditions for disjoint sets of events is an example of the general covering problem described by Michalski (1969) and Michalski and Larson (1978): Given a list of observed events divided into classes, form a general description of each class in terms of the given concepts such that it covers every event in the class and distinguishes this class from the events in other classes. These results are called *discriminant descriptions* and can be used to predict the class membership of any new event. For example, suppose we are presented with examples of two classes as in Figure 6(a). An algorithm known as A^q would generate the descriptions in Figure 6(b) (Michalski, 1969, 1983; Becker, 1985). The description for class A specifies that objects in this group consist solely of clear circles or any kind of triangle. Similarly, class B contains either striped circles or any squares or pentagons. These sets of conditions uniquely determine whether an object belongs to class A or class B.

6.1 The A^q algorithm

The data representation language used in our implementation of A^q is a variable valued logic known as VL_1 (Michalski & Larson, 1978), an example of which is shown in Figure 6(b). Each term in square brackets specifying the value or values of an individual variable is called a *selector* (e.g., [Object = circle]). A conjunction of

(a)



(b)

Class A	**Cover:**	**[Object = circle] [Filler = clear] $\vee$ [Object = triangle]**
Class B	**Cover:**	**[Object = circle] [Filler = striped] $\vee$ [Object = square $\vee$ pentagon]**

Figure 6. A simple classification problem.

Table 3. *Cover* algorithm.

```
FUNCTION Cover (positive_events, negative_events: events): cover;

  While there are still uncovered positive events.

    • choose a seed event from the uncovered positive events
    • generate a star from the seed against the negative events, using the lexical evaluation function
        (LEF) to limit the size of the star.
    • use the LEF to choose the best complex from the star and add it disjunctively to the cover.
    • modify the list of uncovered positive events to reflect the addition of the new complex.

  Return the cover.
```

selectors, represented by writing them together on a single line, is called a *complex* and forms a partial description of a given class set. The entire description of the class is given by a disjunction of such complexes and is called its *cover*. Thus, VL_1 class descriptions are represented in disjunctive normal form (DNF).

The generalization operator in A^q is ExtendAgainst (Michalski, 1983). To extend selector A against selector B, where A and B represent different selectors for the same attribute, generalize the list of possible values for A without including any values B currently possesses. The result of each selector operation is one or more generalized single selector complexes. Variables may have nominal, linear, or structured domains and the operation is defined differently for each.

The A^q method consists primarily of two high-level algorithms. The first is *Cover*, which takes each class set in turn and generates a discriminant cover for the set (Table 3). When a cover is being generated for a class, its events are designated the positive examples and the events from all the other classes are collectively called the negative examples. Cover first selects a single positive event, the *seed*, and passes this to the *Star* algorithm along with the list of negative events. The Star algorithm returns a list of complexes which represent maximally general descriptions of the seed that do not cover any of the negative events. The Cover algorithm then selects the best complex according to a user specified lexicographic evaluation function (LEF), adds this disjunctively to the current cover, and removes from the list of positive events those which are described by the new complex. If any events remain uncovered, a new seed is chosen and the process repeats. A modified version of A^q (called A^q/RU) is used in the current ABACUS implementation; this borrows from Becker's (1985) ExceL system for dealing with exceptions. For a more detailed discussion of these algorithms see Michalski (1983), Becker (1985), and Falkenhainer (1985b).

6.2 A^q in ABACUS

The covers generated by A^q have two possible uses in the ABACUS system. First, the combination of logical conditions with mathematical equations gives the results

Table 4. Analysis of Coulomb's law.

$$Fr^2/q_1q_2 = C_1$$

If [substance = water]	Then C_1 = 8897.352
If [substance = air]	Then C_1 = 111.280
If [substance = silicon]	Then C_1 = 1312.363
If [substance = germanium]	Then C_1 = 1779.015

predictive power. Suppose one were able to only obtain values for $n - 1$ variables of an n variable equation. By knowing which equation should apply prior to evaluating it, one could determine the nth attribute from the other $n - 1$ attributes. Second, the logical conditions often provide additional conceptual meaning for the user. For example, Coulomb's law relating the force of attraction of two charged particles separated by a distance r may be stated as

$$\frac{Fr^2}{q_1q_2} = 4\pi\epsilon$$

where ϵ is defined to be the permitivity of the surrounding medium. Table 4 shows the results obtained by ABACUS when given measurements for the force (F), the distance (r), the charge of particle one (q_1), the charge of particle two (q_2), and the name of the surrounding medium.[4] From the results it can be seen that all of the data obey the same relationship form where the constant in each case is dependent on the surrounding medium. For this example the domain constraints provide an indication that there is some property associated with each substance which affects the electrical attraction of two charged particles. BACON is able to discover a similar form of Coulomb's law by associating an intrinsic property corresponding to permitivity with each nominal variable.

7. Experiments

Some example experiments will now be discussed to show what types of problems ABACUS is able to solve. These experiments investigate:

- gravitational attraction and Stoke's law for viscous fluids
- the law of conservation of kinetic energy
- analysis of chemical compound data

The first two are experiments designed to illustrate the capabilities of the program.

[4] The current implementation of ABACUS actually outputs an equation for each class, as in the example shown in Figure 1. The summarized results given here represent what would be displayed by a postprocessor we have designed for the program.

The data used for these were generated by hand with a knowledge of the correct answer.The final chemistry example represents an experiment run on data provided by members of the University of Illinois Chemistry Department.

7.1 Galilean experiment on free-falling bodies

When Galileo was studying the motion of projectiles, he concluded that the flight of all projectiles could be viewed as two completely separate motions, one in a horizontal direction which is unaffected by the pull of the earth, and the other up and down, controlled by the earth's attraction. His dilemma then was how to describe this vertical component of motion which is so firmly tied to the downward pull of the earth. By dropping various objects through different fluids, he noticed that objects of different weights fell at more nearly the same rate when fluids of lower density were used. From this he deduced that in a vacuum all objects fall at the same rate. Stoke later expanded on this by formulating a law which related the retarding force of a liquid to its viscosity. We presented ABACUS with a set of data to simulate these experiments. The balls dropped came in three sizes, for which there was a rubber ball and a clay ball in each size. The six balls were dropped from rest through three different media, namely glycerol, castor oil, and a vacuum, once each for two different size containers. The experiment was conducted in Death Valley and in Denver, and the temperature was maintained at 20°C at both locations. The measured attributes consist of the height of the container, the mass of the ball, its radius, the duration of the fall, and the velocity with which it strikes the bottom of the container (Figure 7). In addition, the substance through which the ball fell has been noted along with the location of the experiment. Samples of the measurements taken are given in Figure 7.[5] Each ball was dropped once through each medium for both containers at each location for a total of 72 observations.

ABACUS was run twice on the resulting data. First, the default ±2% margin of error was used, resulting in the following observation:

Rule A	*IF*	[substance = Vacuum]
	THEN	$v = 9.8175 \times t$
Rule B	*IF*	[substance = Glycerol]
	THEN	$v \times r = 0.9556 \times m$
Rule C	*IF*	[substance = CastorOil]
	THEN	$v \times r = 0.7336 \times m$

[5] Of course, the correct measurements were calculated by hand for this experiment. The mass of each ball was derived from the chosen radius and the standard densities for rubber and clay. Likewise, the standard viscosity for each substance was used. Gravitational acceleration was chosen to be 9.845 m/s^2 in Death Valley and 9.79 m/s^2 in Denver to reflect the fact that the force of gravity decreases at higher altitudes.

Event$_i$:

velocity:	**18.064 m/s**
radius:	**0.05 m**
mass:	**0.94 kg**
time:	**0.055 s**
height:	**1.0 m**
substance:	**Glycerol**
location:	**DeathValley**

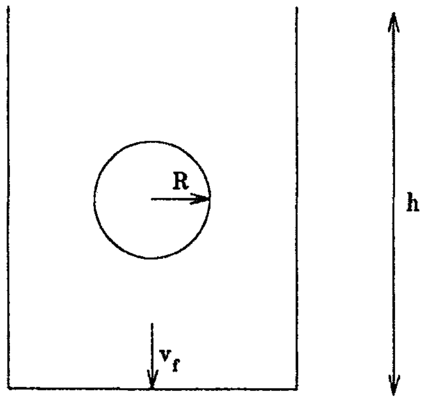


Figure 7. Falling bodies experiment.

It appears that each equation found is dependent upon the medium through which the balls fall. ABACUS was then run a second time using a margin of error of ±0.2% and the following results were reported:

Rule A	*IF*	[location = DeathValley][substance = Vacuum]
	THEN	$v = 9.8453 \times t$
Rule B	*IF*	[location = Denver][substance = Vacuum]
	THEN	$v = 9.7898 \times t$
Rule C	*IF*	[location = DeathValley][substance = Glycerol]
	THEN	$v \times r = 0.9583 \times m$
Rule D	*IF*	[location = Denver][substance = Glycerol]
	THEN	$v \times r = 0.9530 \times m$
Rule E	*IF*	[location = DeathValley][substance = CastorOil]
	THEN	$v \times r = 0.7356 \times m$
Rule F	*IF*	[location = Denver][substance = CastorOil]
	THEN	$v \times r = 0.7315 \times m$

This time each equation is dependent upon both the medium through which the balls fall and the location of the experiment. For the classes corresponding to rules A and B, it would also appear that the behavior is independent of the characteristics of the balls used. Interpreting these findings, we know that an object undergoes a constant acceleration due to earth's gravity and that an object under constant acceleration will change speed proportional to the length of time it undergoes this acceleration. This may be stated as $\Delta v = a\Delta t$ and corresponds to the cases of the balls falling in a vacuum. The constants for these cases simply represent the earth's gravitational acceleration at the two different locations. When we take the resistance of the medium into account, however, as we must do for glycerol and castor oil, the retarding force of the medium becomes involved and is stated by Stoke's law as:

$$F_r = -6\pi\eta r v$$

where η is the viscosity coefficient of the fluid, r is the radius of the ball, and v is the velocity at some point in time. Because of this added force, the object will reach a constant terminal velocity given by

$$v_T = \frac{mg}{6\pi\eta r} \quad \text{or} \quad \frac{v_T r}{m} = \frac{g}{6\pi\eta}$$

where g is the gravitational acceleration. This resembles the equation reported by ABACUS ($v \times r$ = Constant $\times\ m$) for the glycerol and castor oil cases. In these cases, the constant contained values for η and g, explaining why the values reported for rules C–F in the second example were dependent on both the type of liquid (η) and the location (g). If the nominal location and substance variables had been replaced by the numeric variables g and η, respectively, those variables would have shown up explicitly in the equations. This would reduce rules C–F to a single rule with the constant equaling 6π. Knowing when to explicitly represent conceptual constants such as π would be an interesting topic for future research.

This experiment points out a number of properties of ABACUS. First, two different equation forms were discovered, each having only the velocity attribute in common. This demonstrates the program's ability to discover multiple equations for different groups of events, even when variables pertinent to one are irrelevant to another. Second, the necessity and power of the logical preconditions can be seen here. Finally, it points out the problems encountered when working with real numbers, noise, and uncertainties. The results obtained for the ±0.2% case were more interesting and correct than for the ±2% experiment. However, common sense and the presence of noisy data would generally rule out using ±0.2%.

7.2 Conservation of kinetic energy

The law of conservation of energy states that energy can neither be created nor destroyed. Therefore, when two bodies collide, the total energy of the system before the time of collision will be the same as the total energy after the collision. For inelastic collisions, some of this energy is converted to heat during the collision and so, at a macroscopic level, an apparent energy loss is observed. For perfectly elastic collisions, however, the sum of the balls' individual energies, namely their kinetic energies, will remain constant before and after their collisions:

$$\tfrac{1}{2}m_1 v_1^2 + \tfrac{1}{2}m_2 v_2^2 = \tfrac{1}{2}m_1 v_1'^2 + \tfrac{1}{2}m_2 v_2'^2 \tag{7.1}$$

Converting these concepts into an experiment for ABACUS to examine, data were constructed for a series of observations of various objects colliding. The data consisted of seven attributes and 12 events, where the seven attributes consisted of the masses of the two balls, their four corresponding velocities (magnitudes), and a nominal variable which described the observed collision as either elastic or inelastic.[6] For this dataset, ABACUS produced the following results:

Rule A	*IF*	[collision-type = elastic]
	THEN	$m_1(v_1^2 - v_1'^2) = m_2(v_2'^2 - v_2^2)$
Rule B	*IF*	[collision-type = inelastic]
	THEN	No formula was found

An equation equivalent to 7.1 was found to hold for those events corresponding to an elastic collision. No equation could be found for the remaining "inelastic" events. ABACUS was not only able to discover the desired equation, but was also able to specify that the equation only held for elastic collisions. As we will discuss later, ABACUS found this law relatively difficult to discover because it contains subtraction.

7.3 Analysis of chemical compounds

Figure 8 shows the structure of a typical bimetallic coordination compound. The distance between the central metal atoms in such compounds is important to chemists, but it is difficult and expensive to measure. At present, there is no known

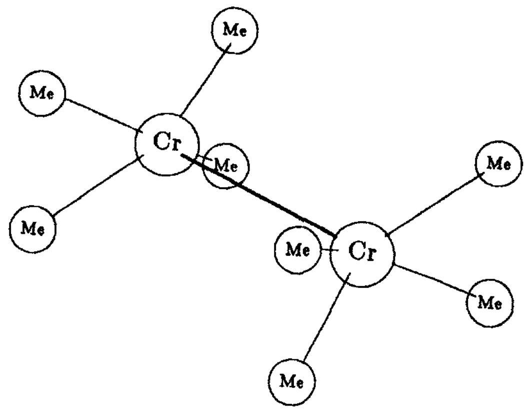


Figure 8. Bimetallic coordination compound in eclipsed conformation.

[6] We recognize that indicating the collision type with a symbolic variable simplifies the data clustering task. The example is still very interesting as it stands, but we would like to develop more powerful clustering techniques, as discussed in the section on future research.

Table 5. Experimental data for bimetallic coordination compounds.

Compound	Metal	Ox	MMdist	Rad	Q	eM	Conf	BO	L1	L2	L3	L4	L5
[Cr(CH3)8]4−	Cr	II	1.980	1.29	−4	12	eclip	4	Me	Me	Me	Me	None
[Mo2C18]4−	Mo	II	2.139	1.40	−4	12	eclip	4	Cl	Cl	Cl	Cl	None
[Mo(CH3)8]4−	Mo	II	2.148	1.40	−4	12	eclip	4	Me	Me	Me	Me	None
Mo2Br4(p-MeC5H4N)4	Mo	II	2.150	1.40	0	12	eclip	4	Br	NR3	Br	NR3	None
Mo2Cl4(p-MeC5H4N)4	Mo	II	2.153	1.40	0	12	eclip	4	Cl	NR3	Cl	NR3	None
Mo2(CH2SiMe3)6	Mo	III	2.167	1.40	0	9	stag	3	CH2SiR3	CH2SiR3	CH2SiR3	None	None
Mo2(NMe2)4Cl2	Mo	III	2.200	1.40	0	9	stag	3	NMe2	NMe2	Cl	None	None
Mo2(NMe2)4Me2	Mo	III	2.201	1.40	0	9	stag	3	NMc2	NMc2	Mc	Nonc	Nonc
Mo2(NMe2)6	Mo	III	2.211	1.40	0	9	stag	3	NMe2	NMe2	NMe2	None	None
Re2Cl6[PEt3]2	Re	III	2.222	1.37	0	12	eclip	4	PEt3	Cl	Cl	Cl	None
Mo2(OCH2CMe3)6	Mo	III	2.222	1.40	0	9	stag	3	OCR3	OCR3	OCR3	None	None
Re2Cl4[P(Et)3]4	Re	II	2.232	1.37	0	14	eclip	4	Cl	PEt3	Cl	PEt3	None
[Re2Cl8]2−	Re	III	2.241	1.37	−2	12	eclip	4	Cl	Cl	Cl	Cl	None
Mo2(OSiMe3)6HNMe2	Mo	III	2.242	1.40	0	11	stag	3	OSiR3	OSiR3	OSiR3	NR3	None
W2[CH2Si(CH3)3]6	W	III	2.255	1.41	0	9	stag	3	CH2SiR3	CH2SiR3	CH2SiR3	None	None
[Re2Br8]2−	Re	III	2.270	1.37	−2	12	eclip	4	Br	Br	Br	Br	None
W2(NMe2)4Cl2	W	III	2.283	1.41	0	9	stag	3	NMe2	NMe2	Cl	None	None
W2(NEt2)4Me2	W	III	2.291	1.41	0	9	stag	3	NEt2	NEt2	Me	None	None
W2(NMe2)6	W	III	2.294	1.41	0	9	stag	3	NMe2	NMe2	NMe2	None	None
W2(NEt2)4I2	W	III	2.296	1.41	0	9	stag	3	NEt2	NEt2	I	None	None
W2Cl2(NEt2)4	W	III	2.301	1.41	0	9	stag	3	Cl	NEt2	NEt2	None	None
W2(NEt2)4Br2	W	III	2.303	1.41	0	9	stag	3	NEt2	NEt2	Br	None	None
W2(OCHMe2)6(pyr)2	W	III	2.332	1.41	0	11	stag	3	OCR3	OCR3	OCR3	NR3	None
Co2(CO)6[P(n-Bn)3]2	Co	O	2.665	1.25	0	17	stag	1	CO	CO	CO	None	PEt3
Mn(CO)8[PEt3]2	Mn	O	2.913	1.37	0	17	stag	1	CO	CO	CO	CO	PEt3
Mn2(CO)10	Mn	O	2.923	1.37	0	17	stag	1	CO	CO	CO	CO	CO
Cr2(CO)10	Cr	−I	2.970	1.29	−2	17	stag	1	CO	CO	CO	CO	CO
Re2(CO)10	Re	O	3.020	1.37	0	17	stag	1	CO	CO	CO	CO	CO
Tc2(CO)10	Tc	O	3.036	1.35	0	17	stag	1	CO	CO	CO	CO	CO
Mo2(CO)10	Mo	−I	3.123	1.40	−2	17	stag	1	CO	CO	CO	CO	CO

way to predict this distance given the values of other attributes, providing a unique challenge for testing the usefulness of ABACUS and thus revealing its strengths and weaknesses.

The compounds are symmetric about the central covalent bond, each side consisting of a primary metal atom and up to five ligand molecules joined by ionic bond to the primary atom. Data for the experiment consisted of the values of 13 attributes for 30 different observed compounds as garnered from various chemical research articles.[7] The full collection of data is reproduced in Table 5, where the compound formula has been added for thoroughness. 'Metal' is the name of the two central metal atoms, 'Ox' is the oxidation state of the metals, 'Rad' is the radius of one metal atom in angstroms, and 'eM' represents the number of electrons per metal. The formal bond order of the covalent bond is given by 'BO', 'MMdist' gives the distance between the metals in angstroms, 'Q' is the total charge of the molecule in units of electron charge, and the conformation (indicating the molecule's alignment) is shown by 'Conf'. In Figure 8, the molecule is in the eclipsed conformation because the ligand molecules of each side line up when viewed from on end. Finally, 'L1' through 'L5' are the names of the ligand molecules.

The experiment began by running ABACUS using all default parameter values. The response was that no relation could be found. A further examination of the output revealed that no nodes were created, thus indicating that either there were no relations in the data or that all proportionality tests returned *Prop*$^?$ and thus no examples could be obtained. The latter situation would occur if the large number of nominal variables was interfering with the numerical relation-finding process. As a result, the program was instructed to ignore all nominal variables when trying to hold variables constant for the proportionality test. This time, nodes were created but still no relation was found to hold for the 40% default constancy criterion. Since actual measured data might contain a reasonable amount of noise, the decision was

Table 6 Initial results with margin of error at 8%

Rule A	*IF*	[L2 = CH2SiR3 v NEt2 v NMe2]
		[L1 = OCR3][L4 = None]
	THEN	MMdist = 0 2502 eM
Rule B	*IF*	[L1 = OSiR3 v PEt3]
		[L2 = Br]
		[L2 = Cl v OCR3][Metal = Re v W]
	THEN	MMdist = 0 1954 eM
Rule C	*IF*	[L2 = CO v Me v NR3 v PEt3]
		[Ox = II]
	THEN	MMdist = 0 1735 eM

[7] Data were collected by J M Hanckel and Theodore L Brown of the University of Illinois Chemistry Department

Table 7. Results using log(BondOrder).

Rule A	*IF*	[OX = III][L4 = Br v Cl v Me v PEt3 v None]
	THEN	MMdist log(BO) = 0.1180 eM
Rule B	*IF*	[L4 = Me v NR3 v PEt3]
		[Q = −4.0000]
	THEN	MMdist log(BO) = 0.1031 eM

made to loosen the default margin of error of 2% to 5% and then again to 8%. Results now began to be reported. A 40% constancy criterion coupled with a 2% margin of error was simply too strict. The results of this run are shown in Table 6. What is most promising about these results is that the same equation was found to hold for all of the data, with only the constant differing. This suggests the discovery of some type of physical phenomenon more strongly than if different equations were uncovered.

After conducting numerous experiments in this manner, the most promising results were shown to the members of the chemistry department for their opinion. While the results looked promising, the conclusion was that these relations did not coincide conceptually with any known physical phenomenon and the margin of error used was far too high for these data. However, the chemists suggested trying the logarithm of the bond order as this type of term appears often in empirical bond order–bond length correlations. Continuing with the experimentation, the program was instructed to replace Bond Order by log(Bond Order), a variety of parameter settings were tried, and the conclusions of Table 7 were obtained. As before, these results looked promising, but the margin of error was still too high.

After further analysis of the data by hand, it was reasoned that perhaps there was too much redundancy in the original data. For example, each metal atom has a unique radius associated with it. Therefore, our chemistry expert suggested that we reduce the number of attributes to 9, consisting of the radius, the metal to metal distance, the bond order, the five ligand names, and a new electrons-per-metal

Table 8. Results using reduced dataset with margin of error at 2%.

Rule A	*IF*	[L1 = CO][Rad = 1.29 ... 1.35]
		[L1 = CO][Rad = 1.4]
	THEN	MMdist = 2.2606 Rad
Rule B	*IF*	[eM* = 18.0]
	THEN	MMdist = 2.1491 Rad
Rule C	*IF*	[L2 = Br v OSiR3 v PEt3]
		[L4 = Cl][Rad = 1.37]
		[Rad = 1.41]
	THEN	MMdist = 1.6279 Rad
Rule D	*IF*	[L2 = Me v NR3]
		[Rad = 1.4][L4 = Cl v None]
	THEN	MMdist = 1.5528 Rad

value (eM*) calculated by adding the bond order to the old value. Since a somewhat new dataset was being used, the margin of error was returned to its default value of 2% and no variables were to be ignored. On the first run, the equations of Table 8 were obtained. Our chemistry expert, Professor Ted Brown from the University of Illinois Department of Chemistry, has judged these equations to be quite interesting as they hold with only a 2% margin of error. He still considers them inconclusive, however, because they need physical explanation. He suggested further analysis to try and uncover an underlying commonality between the compounds of each class which could explain these findings. Nevertheless, this experiment demonstrates that ABACUS is quite useful in analyzing real world data and searching for unknown laws.

8. Discussion of methodology

The experiments presented in Section 7 have shown that ABACUS can potentially be a valuable tool for discovering laws in a variety of domains. This section discusses the current implementation of ABACUS, its limitations, and outlines some problems for future research.

8.1 Analysis of experimental results

ABACUS runs on a Sun Microsystems workstation running FranzLisp under a UNIX environment. A variety of examples have been presented, representing different complexities of equations and preconditions. Table 9 shows a comparison of how complex all of our examples turned out to be. Equations composed solely of multiplication and division have been shown to be quite simple discovery tasks. This is exhibited by the small number of nodes required for all of the examples except the ones for conservation of momentum and kinetic energy. The Coulomb example was more difficult than the ideal gas law because it contained a squared

Table 9. Relevant statistics of the quantitative learning experiments.

Example	Number of Events	Number of Classes	Total Nodes	Equation Discovery Time	Precondition Generation Time	Total Time
Ideal Gas	15	1	3	3	0	3
Momentum	22	3	62	56	12	68
Coulomb	36	5	32	47	27	74
Stoke's Law	72	6	5	18	31	49
Kinetic Energy	12	2	987	850	5	855
Chemistry	30	4	5	3	28	31

NOTE: All times are given in CPU seconds.

term, forcing the proportionality graph search algorithm to exhaustively search the initial, highly interconnected graph. Equations which contain addition or subtraction are significantly more difficult using the methods presented here, but are still quite manageable. From the times, given in CPU seconds, it can be seen that the program was relatively efficient for every example.

Performance is lessened somewhat by large numbers of events. This can be seen by comparing the time required to generate five nodes for the large set of Stoke's law data against the time required to generate five nodes for the smaller chemistry example. Other factors, such as the number of proportionality tests performed, should be considered when comparing these times.

8.2 Limitations

There are a number of problems yet unsolved. Equations that the current ABACUS can discover are limited to those involving multiplication, division, addition, and subtraction operators. The equations are of the form

$$f(x) = \text{Constant}$$

where $f(x)$ is composed solely of user defined variables and operations between them. General polynomials with coefficients cannot be discovered, preventing the discovery of a variety of physical laws. In addition, terms such as sin and log cannot exist unless the user explicitly tells the program to create them.

There are classes of equation forms that cannot be discovered by the methods used in ABACUS. For example, data corresponding to certain parabolas and oscillations often appear to be void of qualitative proportionalities due to the problem of conflicting proportionalities. In addition, the user is not required to supply carefully prepared data in which some attributes are held constant while others are allowed to change. This causes many $Prop^?$ assertions, making the problem more difficult than it would be otherwise.

8.3 Future research

Of major importance is recognizing a desirable term once it is created. When a term is invariant across all events, the task is easy. When only a percentage of events are covered, when should the search algorithm stop? The problem lies in the basic ignorance of the program. What is needed is some form of conceptual knowledge which would enable it to distinguish between conceptually good terms and conceptually useless terms. This became quite evident during testing of the program when occasionally an unexpected answer would be returned which in fact covered more events than the desired equation, but which was mathematically far more complicated. We need to develop a method for the program to consider syntactically desirable equations even when they may not be the most general.

In this work, we investigated what could be accomplished when only minimal information is provided by the user. Previous quantitative discovery systems have required that the user specify which variables are dependent and which are independent. They have also required that all permutations of variable values be given so that *Prop*? assertions will never exist (Langley, 1981). The later condition means that the user must generate a great many more events than needed by ABACUS. In regression analysis, the form of the equation is predefined. In our experiments, however, it soon became evident that the value of ABACUS as a researcher's assistant tool would be enhanced if we allowed additional knowledge to be optional and didn't prevent the user from supplying available knowledge. A simple but useful addition would provide an option for a user to include or exclude specified variables from the equation discovery or precondition generation processes.

The most challenging task we undertook was the discovery of multiple laws for a collection of data. This single decision prevented the use of curve-fitting techniques, eliminating the possibility of discovering a more general class of equations and making many very interesting relationships more difficult to discover. The search strategy employed and the trend detection algorithm used were forced to be quite loose. As a result, the potential search space was increased, irrelevant variables became harder to locate, and conflicting proportionalities became an issue. A possible solution may be to cluster the events prior to invoking the equation discovery module in some manner such that, in each set of events, the events all hold the same set of proportionality assertions. Given this, the more precise approaches such as regression analysis could be taken once again. The method would be based on some form of clustering algorithm, much like the conceptual clustering of Michalski (1980) and Stepp (1984), and might be quite simple, merely forming clustered groups so that all events support the same proportionality assertion. As mentioned earlier, the problem of conflicting proportionalities, such as shown in Figure 1, can be solved for most cases by determining the points where the proportionality between variables changes sign. This may introduce a new problem, that of merging two or more equations into one, as in the case of parabolic or sinusoidal functions.

9. Summary

The methodology of equation discovery and precondition generation used in ABACUS has been presented, analyzed, and illustrated through examples. ABACUS has proved useful for a number of problems in chemistry and physics and the results show that it has been efficient for each learning task.

ABACUS measures well against the criteria we proposed for a quantitative discovery system. It handles irrelevant variables, and is capable of formulating multiple equations for characterizing the data and determining qualitative or logical

preconditions for each equation. The techniques of variable combination through search, as used in the early BACON programs, have been considerably improved upon by analyzing the specific search characteristics of the domain. The current version of ABACUS falls short on two points. It is relatively limited in the kinds of equations which can be discovered, and it occasionally suffers from the computational cost of search.

The work presented here is unique in several ways. First, no prior work has addressed the real-world issue of discovering multiple equations to describe different aspects of a physical situation. Even for cases where it appears that a single physical phenomenon is being observed, as in the falling bodies example, different physical situations may exist requiring different equation forms to describe them. Previous programs have been unable to discover different equations for different subsets of the given events. Second, the explicit generation of logic-style preconditions for the discovered equations is novel. When different physical situations exist for what appears to be the same phenomenon, preconditions determine when each equation applies. Finally, new search techniques for equation discovery have been created. In conclusion, the ABACUS system seems to be a useful new tool in the analysis of experimental data.

Acknowledgments

We would like to thank Theodore L. Brown and J. M. Hanckel from the University of Illinois Chemistry Department for providing the data used in the chemical compound analysis experiments, interpreting the results, and supplying very valuable comments about the observed methodology. The authors thank Pat Langley for many valuable comments and criticisms on an earlier version of this paper.

This work was supported in part by the Defense Advanced Research Project Agency under grant N00014-K-85-0878, the National Science Foundation under grant DCR 84-06801, and the Office of Naval Research under grant N00014-82-K-0186.

References

Aho, A.V., Hopcroft, J.E., & Ullman, J.D. (1974). *The design and analysis of computer algorithms*. Menlo Park, CA: Addison-Wesley.

Becker, J.M. (1985). *Inductive learning of decision rules with exceptions: Methodology and experimentation*. Master's thesis, Department of Computer Science, University of Illinois, Champaign-Urbana, Illinois.

Chatterjee, S., & Price, B. (1977). *Regression analysis by example*. New York: John Wiley and Sons.

Condon, E.U., Odishaw, H. (Eds.). (1967). *Handbook of physics* (2nd ed.). New York: McGraw-Hill.

Daniel, C., & Wood, F.S. (1971). *Fitting equations to data*. New York: Wiley-Interscience, John Wiley & Sons.

de Kleer, J. (1975). *Qualitative and quantitative knowledge in classical mechanics*. Master's thesis (TR-

352), Massachusetts Institute of Technology, Cambridge, Massachusetts.

Dietterich, T., & Michalski, R.S. (1981). Inductive learning of structural descriptions: Evaluation criteria and comparative review of selected methods. *Artificial Intelligence, 16* (3), 257–294.

El-Shafei, N. (1986). *Quantitative discovery and reasoning about failure mechanisms in pavement.* Unpublished manuscript, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Falkenhainer, B.C. (1984). *ABACUS: Adding domain constraints to quantitative scientific discovery* (Technical Report UIUCDCS-F-84-927, ISG 84-7). Department of Computer Science, University of Illinois, Champaign-Urbana, Illinois.

Falkenhainer, B.C. (1985a). Proportionality graphs, units analysis, and domain constraints: Improving the power and efficiency of the scientific discovery process. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 552–554). Los Angeles, CA: Morgan-Kaufmann.

Falkenhainer, B.C. (1985b). *Quantitative empirical learning: An analysis and methodology.* Master's thesis (UIUCDCS-F-85-947, ISG 85-16), Department of Computer Science, University of Illinois, Champaign-Urbana, Illinois.

Forbus, K.D. (1984). *Qualitative process theory.* Doctoral dissertation (TR-798), Department of Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Hajek, P., & Havranek, T. (1978). The GUHA method: Its aims and techniques. *International Journal on Man Machine Studies, 10,* 3–22.

Huntley, H.E. (1952). *Dimensional analysis.* London: MacDonald.

Kokar, M. (1981). A procedure of identification of laws in empirical sciences. *Systems Science, 7,* 32–41.

Kokar, M. (1986). Determining arguments of invariant functional descriptions. *Machine Learning, 1,* 403–422.

Kowalik, J.S. (1986). *Coupling symbolic and numerical computing in expert systems.* Amsterdam: North-Holland.

Langhaar, H.L. (1951). *Dimensional analysis and theory of models.* New York: John Wiley & Sons.

Langley, P. (1979). Rediscovering physics with BACON.3. *Proceedings of the Sixth International Joint Conference on Artificial Intelligence* (pp. 505–507). Tokyo, Japan: Morgan-Kaufmann.

Langley, P. (1981). Data-driven discovery of physical laws. *Cognitive Science, 5,* 31–54.

Langley, P., Bradshaw, G.L., & Simon, H.A. (1981). BACON.5: The discovery of conservation laws. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence* (pp. 121–126). Vancouver, BC, Canada: Morgan-Kaufmann.

Langley, P., Bradshaw, G.L., & Simon, H.A. (1983a). Rediscovering chemistry with the BACON system. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Palo Alto, CA: Tioga.

Langley, P., Bradshaw, G.H., Simon, H.A., & Zytkow, J. (1983b). Mechanisms for qualitative and quantitative discovery. *Proceedings of the International Machine Learning Workshop.* Monticello, Illinois.

Langley, P., Zytkow, J., Simon, H.A., & Bradshaw, G.L. (1986). The search for regularity: Four aspects of scientific discovery. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Michalski, R.S. (1980). Knowledge acquisition through conceptual clustering: A theoretical framework and an algorithm for partitioning data into conjunctive concepts. *Policy Analysis and Information System, 4,* 219–244.

Michalski, R.S. (1983). A theory and methodology of inductive learning. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Palo Alto, CA: Tioga.

Michalski, R.S., & Larson, J.B. (1978). *Selection of most representative training examples and incremental generation of VL1 hypotheses: The underlying methodology and the description of programs ESEL and AQ11* (Technical Report UIUCDCS-R-78-867). Department of Computer Science, University of Illinois, Champaign-Urbana, Illinois.

Mitchell, T.M. (1982). Generalization as search. *Artificial Intelligence, 18*, 203–226.

Roller, D.E., & Blum, R. (1981). *Physics: Mechanics, waves, and thermodynamics* (Vol. 1). San Francisco, CA: Holden-Day.

Stepp III, R.E. '(1984). *Conjunctive conceptual clustering: A methodology and experimentation.* Doctoral dissertation (UIUCDCS-R-84-1189), Department of Computer Science, University of Illinois, Champaign-Urbana, Illinois.

Weast, R.C. (Ed.). (1984). *CRC handbook of chemistry and physics* (65th ed.) CRC Press.

Zagoruiko, N.G. (1976). Empirical prediction algorithms. In J.C. Simon (Ed.), *Computer oriented learning process* (pp. 581–595). Leyden: Noordhoff.

Zagoruiko, N.G., Elkina, V.N., & Lbov, G.S. (1985). *Algorithms for revealing empirical laws.* Nauka Publishing House, Siberian Division of the Soviet Union Academy of Sciences.