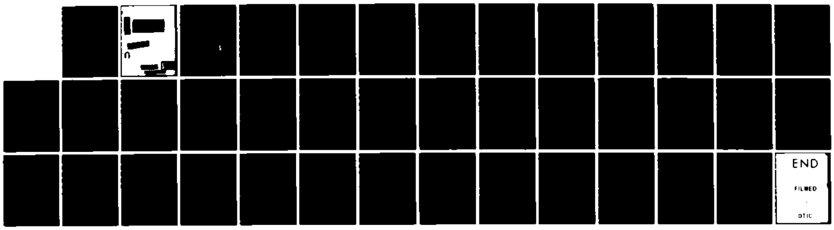
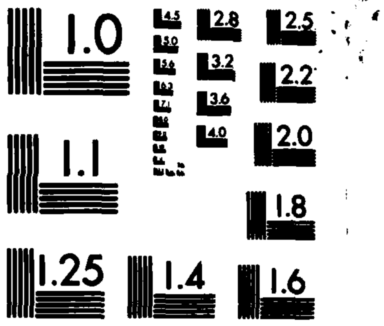
AD-A145 939 THE SEARCH FOR REGULARITY: FOUR ASPECTS OF SCIENTIFIC DISCOVERY(U) CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST P LANGLEY ET AL. 01 SEP 84 1/1

UNCLASSIFIED CMU-RI-TR-84-20 N00014-84-K-0345 F/G 6/4 NL




1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

# The Search for Regularity: Four Aspects of Scientific Discovery

Pat Langley
Jan Zytkow
Herbert A. Simon
Gary L. Bradshaw

CMU-RI-TR-84-20

12

# The Search for Regularity: Four Aspects of Scientific Discovery

Pat Langley
Jan Zytkow
Herbert A. Simon
Gary L. Bradshaw

CMU-RI-TR-84-20

The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213 USA

September 1, 1984

This research was supported by Contract N00014-84-K-0345 from the Information Sciences Division, Office of Naval Research. Approved for public release; distribution unlimited. Reproduction in whole or part is permitted for any purpose of the United States Government.

To appear in R. Michalski, J. Carbonell, and T. Mitchell (Eds.), *Machine Learning*, Volume 2. Palo Alto, California: Tioga Press.

unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER Technical Report No. 3 | 2. GOVT ACCESSION NO. A145 939 | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) The Search for Regularity: Four Aspects of Scientific Discovery | | 5. TYPE OF REPORT & PERIOD COVERED Interim Report 3/84-8/84 |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s) Pat Langley, Jan Zytkow Herbert A. Simon, Gary L. Bradshaw | | 8. CONTRACT OR GRANT NUMBER(s) N00014-84-K-0345 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS The Robotics Institute Carnegie-Mellon University Pittsburgh, PA 15213 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS Information Sciences Division Office of Naval Research Arlington, Virginia 22217 | | 12. REPORT DATE 1 September, 1984 |
| | | 13. NUMBER OF PAGES 34 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) | | 15. SECURITY CLASS. (of this report) unclassified |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

To appear in R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.), Machine Learning, Volume 2. Palo Alto, CA: Tioga Press, 1985.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

| | |
|---|---|
| scientific discovery | heuristic search |
| empirical laws | theory of acids and bases |
| structural models | theory of phlogiston |
| qualitative laws | atomic theory |

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

OVER

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20.

artificial intelligence

## Abstract

Scientific discovery is a complex activity involving many different components. Our interest in discovery has led us to construct four AI systems that address different facets of this process. BACON.6 focuses on discovering empirical laws that summarize numerical data. This program searches a space of data and a space of numerical laws, and includes methods for postulating intrinsic properties and noting common divisors. GLAUBER is concerned with discovering laws of qualitative structure, such as the hypothesis that acids react with alkalis to form salts. It searches the space of qualitative laws, using evaluation functions to focus attention on laws covering the greatest number of observed facts. STAHL attempts to determine the components of substances involved in reactions, and has been used to model the reasoning that led to the phlogiston theory. This system searches through the space of componential models, using heuristics to make plausible inferences. The final system, DALTON, is concerned with formulating structural models of chemical reactions. It searches the space of possible models, considering simple models before more complex ones and using a conservation assumption to constrain possibilities. While each of these discovery systems is interesting in its own right, we are also exploring ways in which the systems can interact to help direct each other's search processes.

## Table of Contents

## 1. Exploring the Scientific Process

Science is a multi-faceted process, concerned with both the collection of data and its explanation. Within these two basic components, we find additional subdivisions. The first process ranges from exploratory data-gathering to the design of specific experiments to test explicit hypotheses. Similarly, the explanatory process ranges from the induction of simple empirical laws to the formulation of complex structural and process models. These components are not independent, since the relation between data and theory is all-important in science. Still, the relations among the various components are complex, and if we ever hope to understand the scientific process, we must resort to powerful methods.

In this paper, we apply the methods of Artificial Intelligence (AI) to explore the processes of scientific discovery. Our goal is not to explain historical details, though the history of science is fascinating and we will certainly draw upon it in our efforts. Rather, we hope to understand the processes by which scientific discoveries *could* have been made; our goal is to develop methods that are *sufficient* for making such discoveries. To this end, we will draw upon the AI technique of implementing theories as running computer programs. Thus, we will devote much of the paper to describing particular AI programs and their behavior in specific domains.

One of the central insights of AI is that intelligence involves the ability to *search*, and the ability to direct that search in profitable directions. Search involves the exploration of some space of possibilities, which Newell and Simon [1] have called a *problem space*. A problem space is defined by two components: (1) one or more *initial* states from which search begins; and (2) one or more *operators* for generating new states from existing ones. Taken together, these components determine a set of states that can be systematically searched. In order to search such a space, one also needs some search control scheme – which directs search down one path or another – and some test – which determines when the goal state has been reached. The notion of problem spaces is important for each of our discovery systems, and we will describe each system in terms of its search characteristics.

We have organized the paper around four AI systems that address different aspects of the discovery process. First we describe BACON, a system that is concerned with discovering empirical laws of a quantitative nature. We will begin with BACON since it is the first discovery system we constructed, and many readers may have some familiarity with it. More important, our recent work has been largely motivated by BACON's limitations, so a consideration of the system's capabilities and limits will lay a solid foundation for the rest of the paper. After this we describe GLAUBER, a system that is also concerned with empirical laws, but in this case, laws having a qualitative form. Next we examine STAHL, a program that infers the components of substances from reactions, followed by DALTON, a system that constructs simple structural models. Since these systems address complementary aspects of the discovery process, we close by discussing some possible interactions between the programs, and the possibility of constructing an integrated discovery system.

## 2. Discovering Quantitative Empirical Laws

At the very end of the 18th Century and the beginning of the 19th, three fundamental discoveries were made that shaped the directions of chemical research for several generations thereafter. The first of these was Proust's (1799) statement of the law of constant proportions. Proust conducted a painstaking analysis of chemical compounds, finding that the ratio of the combining weights of the constituent elements was always constant for a particular compound. The second fundamental advance was Dalton's (1804) introduction of the law of multiple proportions. This law asserts that when two elements combine to form several different compounds, the ratios of their combining weights are always small integer multiples of one another. The third advance was Gay-Lussac's (1809) discovery of the law of combining volumes for gaseous reactions, which states that gases combine with each other in very simple ratios by volume.

These three discoveries provided the foundation for a quantitative theory of chemical reactions, and ultimately led to the determination of relative atomic weights. To some extent, Dalton's and Guy-Lussac's laws were motivated by an atomic hypothesis, but there were strong empirical components to the discoveries as well. Although Proust's law might be dealt with using traditional curve-fitting techniques, the other laws involve more complex relations. Thus, the history of early chemistry provides a challenging domain for testing AI methods for empirical discovery. Below we describe a discovery system that focuses on quantitative discovery, and examine its approach to finding these chemical laws, as well as other laws from the history of science.

**Table 1. BACON's method viewed as search through a data space.**

**Initial state: the null combination [ ]**

**Goal state: a complete experimental combination of independent values**

**Intermediate state: a partial combination of independent values**

**Operators:**
**Specify-value: Specifies the value of an undetermined independent value**

**Heuristics/Evaluation functions:**
**None: Search is exhaustive**

**Search control: Exhaustive depth-first search with backtracking; generates all goal states**

## 2.1. Searching the Space of Data

We have explored the process of quantitative discovery through BACON.6, the sixth in a line of programs named after Sir Francis Bacon (1561-1626). The system is given a set of independent and dependent variables, and based on data it gathers, the program generates empirical laws that relate these variables to each other. In order to achieve this goal, BACON varies one of the independent variables, looking for relations between that variable and the dependent variables. Once a functional relation has been found, the parameters in that function are given the status of dependent variables at a higher level of description. The system then repeats this process with a different value for the second independent variable, arriving at a new set of parameters. When all values of the second independent variable have been considered, BACON has a set of higher level dependent values (based on the parameters) associated with each of the independent values. The system finds a numeric relation between these variables, and again the parameters become dependent values at the next higher level of description. This process continues until all the independent variables have been incorporated into a complex quantitative relationship.

BACON can be viewed as searching two distinct problem spaces — the space of *data* and the space of *laws*. These searches interact in a complex manner, but before we examine this interaction, let us examine each of the search schemes independently, starting with search through the data space. As we have noted, BACON is provided with a set of independent variables, along with possible values for each variable. Using these values, the system generates a complete factorial design involving all combinations of independent values, and then examines the values of the known dependent variables for each combination. BACON's generation of all independent combinations can be viewed in terms of search, with states containing partially specified experimental combinations. The initial state has no independent values specified, while goal states have values for all of the independent terms. The operator for moving through this space inputs a partially specified experimental combination, and decides on the value for one of the unspecified variables. Search control is depth-first, but since many combinations must be generated, the system must backtrack and explore many different paths.

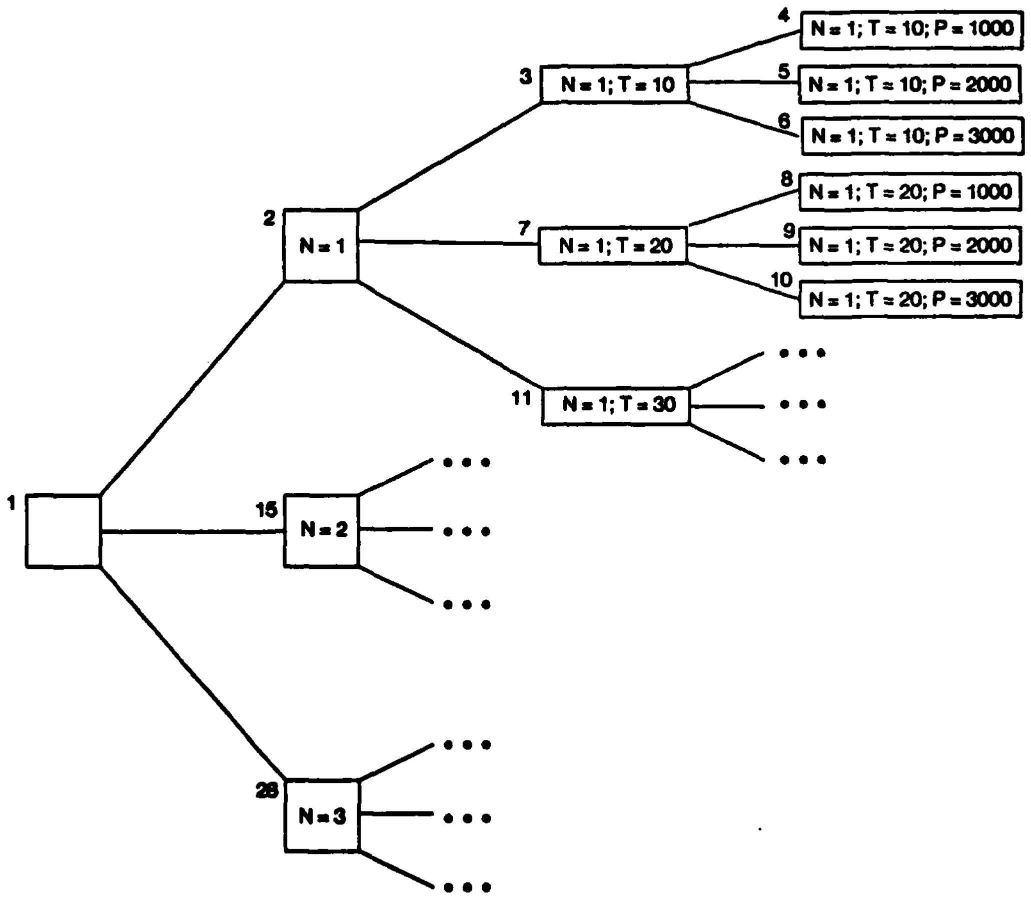


**Figure 1. BACON's search through the space of data.**

For instance, suppose BACON is given three independent variables – the pressure P on a gas, the temperature T of that gas in degrees Celsius, and the quantity N of the gas – along with the single dependent variable V, the volume of the gas. Further suppose that BACON is told to examine N with values 1, 2, and 3, T with values 10, 20, and 30, and P with values 1000, 2000, and 3000. In order to generate an experimental combination, the system begins with an initial state in which no values have been specified, which we may represent as [ ]. Next, the SPECIFY-VALUE operator applies, generating a new state in which the value of N is determined, say [N=1]. Upon its next application, the operator generates a third state in which the value of P is given, say [N=1, T=10]. When BACON applies the operator a third time, the complete experimental combination [N=1, T=10, P=1000] is generated, and the program can examine the volume associated with this combination.

However, if BACON is to gather sufficient data on which to base its laws, it must continue the search. Accordingly, the system backs up to the previous state [N=1, T=10] and applies the operator with different arguments, generating the second goal combination, [N=1, T=10, P=2000]. This allows a second value of the volume to be observed and associated with an experimental combination. At this point, the system again backtracks to [N=1, T=10], and then generates a third goal state, [N=1, T=10, P=3000], thus gathering a third observation of the volume. Having exhausted the potential values of T, BACON then proceeds to back up two steps to [N=1]. From here it generates the states [N=1, T=20] and finally [N=1, T=20, P=1000] another complete experimental combination. BACON continues in this fashion until it has generated all experimental combinations of the independent values it was given, and observed the volumes associated with each combination. Figure 1 shows the tree that results from this search through the space of data; the numbers on each state represent the order in which that state is generated.[1]

**Table 2.** BACON's method viewed as search through the space of laws.

**Initial states:** sets of parameters consisting of 1, 0, and -1

**Goal state:** a set of parameters that maximally predict the observed data

**Intermediate states:** sets of parameters that account for some of the data

**Operators:**
  **Add/Subtract:** Adds or subtracts from one parameter value

**Evaluation function:**
  Select states that lead to higher correlations, thus better predicting the data

**Search control:** Hill-climbing using a beam search

## 2.2. Searching the Space of Laws

Now let us turn to BACON.6's method for searching the space of numeric laws. Given a set of independent values and a corresponding set of dependent values, the system attempts to find one or more laws that predict the observed values as accurately as possible. In order to achieve this goal, BACON requires some information about the *form* that plausible laws may take. For instance, for the independent variable x and the dependent variable y, the user may tell the program to consider laws having the form $y = ax^2 + bx + c$, as well as those with the form $\sin(y) = ax + b$. These forms define the space of laws that BACON will explore in its attempt to summarize the observed data.[2]

Given a set of forms, BACON generates a set of initial states from which to begin the search. This is done by inserting the abstract parameters in each form with the values 1, 0, or -1. For simplicity, let us consider only the form $y = ax + b$ and examine the resulting initial states. In this case, there $3^2 = 9$ possible initial states: [a=1, b=1], [a=1, b=0], [a=1, b=-1], [a=0, b=1], [a=0, b=0], [a=0, b=-1], [a=-1, b=1], [a=-1, b=0], and [a=-1, b=-1]. These parameters are chosen because they are evenly distributed throughout the space of parameters, so that the best set of parameters should be near one of them. Starting from these idealized parameters, BACON attempts to determine the optimum state through a process of successive approximation.

[1]An earlier version of BACON [2] was capable of modifying this search based on discoveries it had made. The current system does not include this ability.

[2]Earlier versions of BACON were restricted to particular forms. For instance, BACON.5 [2] only considered laws of the form $y^i = ax^2 + bx + c$, where i took on small integral values, and thus was less flexible than the current system.

BACON.6 employs a single operator for moving through the space of parameters. This operator accepts one of the current sets as input, and generates a new parameter combination by adding or subtracting some number from one of the existing values. The amount that is added or subtracted decreases as the system's search progresses. For instance, the system begins by adding/subtracting 0.5 from the various values. On the second step, this amount is reduced to 0.25, and so on. BACON's strategy for exploring the parameter space is best described as a beam-search version of hill-climbing. At the outset, the N best states are selected for further attention, and the remainder are abandoned. The addition/subtraction operator is then applied to these N states in all possible ways, generating a new set of M states. Of these N + M states, the N best are selected (some of the originals may be retained), and the process is repeated. When none of the new states show any improvement over the preceding states, the search is terminated.

In selecting some states in favor of others, BACON considers the ability of each parameter set to predict the observed values. In order to estimate this ability, the parameter values are substituted into the form of the law, and the correlation between the observed independent and dependent values is computed. A high correlation means that the parameters predict the data well, while a low correlation implies that the state's predictive ability is poor. Since correlations are insensitive to absolute values, only the *relative* values of the parameters are important. It is for this reason that the initial values of 1, 0, and -1 were able to "cover" the space of parameters. In any case, this evaluation function is used to direct search towards sets of parameters that account for as much of the data as possible.

Since this search strategy only uses the data to *test* hypotheses, and not to generate them, it is robust with respect to numerical noise. BACON.6 is guaranteed to find some law that summarizes regularity in the data, even if this regularity is only partial. Of course, when the data are very noisy, there may not be one set of parameters (or even one form of law) that is clearly superior to its competitors. In such cases, the program returns a number of laws. One of BACON's interesting features is that the system carries out the same amount of search regardless of the amount of noise in the data.

## 2.3. Relation Between the Search Methods

Now that we have examined BACON's two search schemes in isolation, it is time to consider their relation to one another. Basically, the system's search for laws is *embedded* within its search for data. To understand this statement, let us return to Figure 1, which presents the order in which BACON gathers its data. Consider the N leftmost terminal nodes, [N=1, T=10, P=1000], [N=1, T=10, P=2000], and [N=1, T=10, P=3000]. For each of these combinations, the system observes some value of the dependent volume V. When all three values have been noted, BACON attempts to find a law relating them to the three values of the pressure P, using the search strategy just described. The result of this search is one or more parameters (let us assume that one law is obviously better than all others), and these are stored at the next higher state in the data search tree. For instance, for P = 1000, 200, and 3000, the observed values for V would be 2.36, 1.18, and 0.78. For these data, the form $V^{-1} = aP + b$ gives the best fit, with the parameter values a = 0.000425 and b = 0. The value for a is stored with the state [N=1, T=10] for future use; however, the system treats 0 as a special value, so the result for b would not be stored.

Upon observing a second set of values, BACON attempts to find a second law. For the experimental combinations [N=1, T=20, P=1000], [N=1, T=20, P=2000], and [N=1, T=20, P=3000], the system finds the value 2.44, 1.22, and 0.81 for the volume. Again the form $V^{-1} = aP + b$ proves useful, this time with the values a = 0.000410 and b = 0, and again these values are stored at a higher state, in this case [N=1, T=20]. Very similar events occur when the value of T is 30, giving the parameter values a = 0.000396 and b = 0, which are stored with [N=1, T=30]. At this point, BACON has three sets of values for the higher level dependent terms a and b. Moreover, these values are stored with the abstracted combinations [N=1, T=10], [N=1, T=20], and [N=1, T=30]. Given the values 10, 20, and 30 for T, the values 0.000425, 0.000410, and 0.000396 for a, the program attempts to find a law relating these two terms. In this case, it finds the form a = cT + d to best summarize the data, with c = 8.32 and d = −2271.4. These values are stored with the next higher state in the data tree, [N=1], for future use.

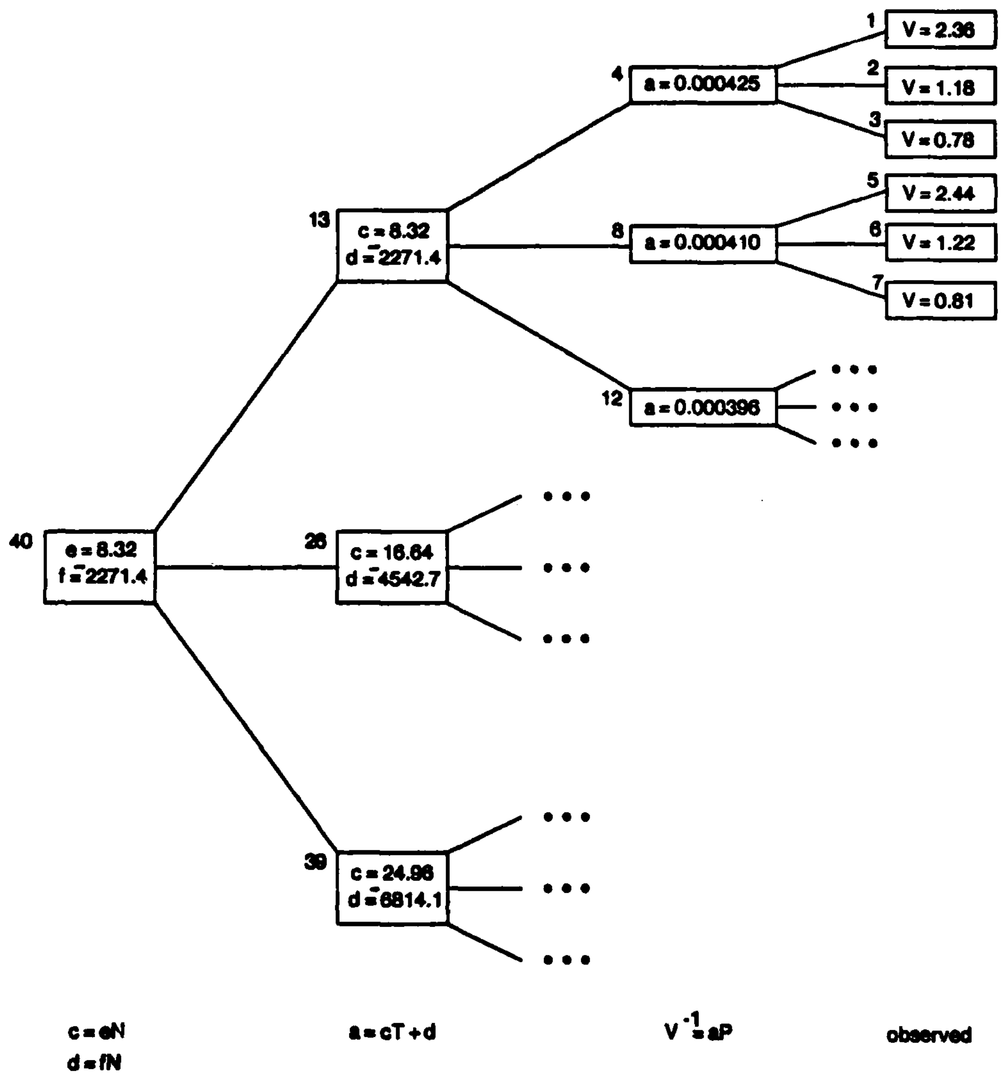


Figure 2. BACON's rediscovery of the ideal gas law.

This process is continued as more data are gathered. First BACON finds three additional laws relating the variables P and V. Based on the resulting parameter values, the form a = cT + d is again found to be useful, this time with c = 16.64 and d = −4542.7. These higher level dependent values are stored with the state [N=2]. Similar steps lead to three more laws of the form $V^{-1}$ = aP + b, and then to a third law of the form a = cT + d. This time BACON finds the best fit with c = 24.96 and d = −6814.1, and stores these values with [N=3]. Now the system has three values of N, along with three associated values of both c and d. For each of these dependent terms, BACON searches the space of laws, arriving at the two laws c = eN and d = fN, with e = 8.32 and f = −2271.4. These two parameter values are stored at the initial data state [ ], and represent invariant parameters that are not conditional on any independent terms. By substituting these values into the forms found at each level in BACON's search, we arrive at the relation $V^{-1} = (8.32NT - 2271.4N)^{-1}P$. This expression can be transformed into $PV = 8.32NT - 2271.4N$ if we divide through by P and invert the equation. If we then factor out 8.32N on the right side of the relation, we arrive at $PV = 8.32N(T - 273)$, which is the standard form of the ideal gas law. Note that in some sense, BACON has determined that the Celsius temperature scale is insufficient for describing the relation between the four terms, and has effectively introduced the Kelvin scale by subtracting 273 from the observed Celsius values.

From this example, we see that BACON carries out as many searches through the law space as there are non-terminal states in the data space. Figure 2 summarizes the parameter values resulting from each of these searches, along with the data states at which they are stored. The number next to each state represents the order in which that law was discovered. Note that this order is different from the order in which the data space itself was searched. In an important sense, the search for data provides structure to BACON's search for laws, since it provides both direct observations and a place to store parameters so they can be used as data at later stages. This process is somewhat similar to Rosenbloom's model of the chunking process [3]. In this cognitive simulation, a goal hierarchy provides the top-down control that determines the *types* of chunks that should be formed. However, a data-driven learning mechanism determines the particular chunks that are acquired from the bottom up. Thus, BACON's search through the data space can be viewed as providing top-down constraints on the types of laws that will be discovered (e.g., which variables are related), while the system must still search through the resulting law space to determine the particular laws that best summarize the data.

We should mention in passing that once BACON discovers that a particular form of law is useful in one context, it uses that information to constrain search in similar contexts. For instance, when the system finds that only the form $V^{-1} = aP + b$ is useful when $[N=1, T=10]$, it considers only this form when $[N=1, T=20]$, $[N=1, T=30]$, and so forth. In addition, since it found $b = 0$, this parameter was removed from the form, leaving the simplified expression $V^{-1} = aP$. In other words, BACON redefines its problem space (i.e., the space of laws) in the light of its previous experience, so that considerably less search results. Now that we have examined BACON's basic methods for discovering empirical laws, let us examine some additional methods that let it deal with the chemical domain.

## 2.4. Intrinsic Properties and Common Divisors

While BACON's basic methods are useful for discovering relations between numerical terms, they cannot be used to relate *nominal* or symbolic independent terms to numeric dependent variables, and this is precisely the problem that confronted the early chemists. For instance, the independent variables in Proust's, Dalton's, and Gay-Lussac's chemical experiments were the elements or compounds involved, while the dependent variables were numerical measures such as weight or volume. In such cases, BACON defines *intrinsic properties* that take on numeric values, and then associates these properties with the nominal terms.

Let us consider the role of intrinsic properties in BACON's rediscovery of the early chemical laws. Given control over the substances entering and resulting from a reaction, as well as the weight of the first substance that is used, the system gathers data like those shown in Table 3. Upon varying the amount of oxygen used to form nitric oxide (NO), the program discovers that the two weights $w_1$ and $w_2$ are linearly related with a slope of 1.14 and an intercept of zero. Upon varying the output of the reaction, BACON.6 then examines the weight relations for the compound nitrous oxide ($N_2O$). In this case, the law is also linear, but the slope has changed to 0.57. A similar result is obtained when the system examines the values for nitrogen dioxide, and in this case the slope is 2.29.

The slopes that BACON.6 finds in these experiments are closely related to the weight ratios found by Proust. Having found these ratios, the program stores its results at a higher level of description, as shown in Table 4, and treats these summaries as data. The table also includes the results obtained for two reactions of oxygen and carbon. In this case, the system finds three nominal independent variables and a single numeric dependent variable, so it defines an intrinsic property (say p) whose values are associated with the three nominal values under which they occur. Thus, the value of p for the triple nitrogen/oxygen/nitric oxide would be set to 1.14, the value for nitrogen/oxygen/nitrous oxide would be 0.57, and the value for nitrogen/oxygen/nitrogen dioxide would be 2.29. As stated, these intrinsic values simply store an already known fact, and in this sense they are tautological. However, they can be retrieved in future experiments involving the same chemicals, and used to make predictions or to discover new empirical laws.

Table 3. Determining the combining weights for reactions.

| $ELEMENT_1$ | $ELEMENT_2$ | COMPOUND | $W_1$ | $W_2$ | $W_2/W_1$ |
|---|---|---|---|---|---|
| NITROGEN | OXYGEN | NO | 1.0 | 1.14 | 1.14 |
| NITROGEN | OXYGEN | NO | 2.0 | 2.28 | 1.14 |
| NITROGEN | OXYGEN | NO | 3.0 | 3.42 | 1.14 |
| NITROGEN | OXYGEN | $N_2O$ | 1.0 | 0.57 | 0.57 |
| NITROGEN | OXYGEN | $N_2O$ | 2.0 | 1.14 | 0.57 |
| NITROGEN | OXYGEN | $N_2O$ | 3.0 | 1.71 | 0.57 |
| NITROGEN | OXYGEN | $NO_2$ | 1.0 | 2.28 | 2.28 |
| NITROGEN | OXYGEN | $NO_2$ | 2.0 | 4.56 | 2.28 |
| NITROGEN | OXYGEN | $NO_2$ | 3.0 | 6.84 | 2.28 |

As we have seen, Proust's insight about combining weights laid the groundwork for Dalton's law of multiple proportions. This law states that in cases where two elements combine to form *different* compounds, the ratios of their combining weights were always small integer multiples of one another. BACON includes a method that lets it discover just such a relation in the data from Table 4. This method, which operates whenever the system defines a new intrinsic property, examines the values of the new property to see if they (or their inverses) have a common divisor. This technique is especially useful when intrinsic values are associated with multiple nominal values, as often occurs in chemistry. We have described both the intrinsic property and common divisor methods at length in earlier papers [4, 5].

In this case, BACON notes that 1.14, 0.57, and 2.28 have the common divisor 0.57, and would replace these intrinsic values with their corresponding integers 2, 1, and 4. In addition, the program defines a higher level intrinsic property based on the divisors it finds in different situations, and associates the divisors with those cases. Thus, the common divisor 0.57 would be associated with the nitrogen/oxygen pair, while the divisor 1.33 would be associated with carbon and oxygen. These relations are formally equivalent to Dalton's law of multiple proportions. BACON takes a similar path in discovering Gay-Lussac's common divisors for combining volumes, and has even arrived at the correct relative atomic weights for hydrogen, oxygen, and nitrogen from data similar to those in Table 3. Thus, BACON's discovery mechanisms account for the major quantitative laws found by chemists in the early 19th Century. Note that neither the intrinsic property method nor the common divisor method involve any significant search themselves. Rather, their role is to transform symbolic data into numeric data, so that BACON's law-finding method can be used to discover relationships.

## 2.5. Comments on BACON.6

In the preceding pages, we have described BACON's methods for gathering data, discovering numeric laws, and postulating new properties. All in all, BACON provides an interesting and useful account of the discovery of quantitative empirical laws. However, the system leaves some important questions unanswered. For example, how do scientists decide which variables to employ in their experiments? Similarly, how do they use their newly discovered laws once they have been found? In BACON, the relevant variables are provided by the programmer, and the laws are simply printed on a terminal screen. One can imagine a version of BACON with an improved user interface, serving as a scientist's aide in analyzing data, and fulfilling a useful function while still requiring its user to design its input and interpret its output. This is one direction in which the system might be extended, and such an interactive version could be very useful in some areas of science.

Table 4. Noting common divisors for chemical reactions.

| $ELEMENT_1$ | $ELEMENT_2$ | COMPOUND | $W_2/W_1$ | P | $W_2/W_1P$ |
|---|---|---|---|---|---|
| NITROGEN | OXYGEN | NO | 1.14 | 2 | 0.57 |
| NITROGEN | OXYGEN | $N_2O$ | 0.57 | 1 | 0.57 |
| NITROGEN | OXYGEN | $NO_2$ | 2.28 | 4 | 0.57 |
| CARBON | OXYGEN | CO | 1.33 | 1 | 1.33 |
| CARBON | OXYGEN | $CO_2$ | 2.66 | 2 | 1.33 |

However, if one's goal is to understand the nature of scientific discovery, then a deeper answer to the above questions is required. For instance, we know from the history of science that empirical laws eventually lead to theories and explanations, and BACON has little to say about such aspects of discovery. We also know that even vague theories can have important impacts on the data one gathers. This suggests that we will find answers to both questions only by studying other facets of the discovery process. Although constructing AI models of these components would undoubtedly be interesting even in isolation, the true advantage will come from exploring the interrelations among different forms of discovery. Our long-range goal, then, should be to understand components of the discovery process whose outputs can be used as BACON's inputs, and to uncover other components that can employ BACON's outputs as their inputs. In the remainder of the paper, we focus on three different models of discovery that we have constructed to this end, and we close with some speculations on possible interactions among the various systems.

## 3. Discovering Qualitative Empirical Laws

In the history of science we find that the discovery of quantitative laws is generally preceded by the discovery of qualitative relations. Thus, early physicists noted that colliding objects tended to change velocities before they determined the exact form of this relationship. Similarly, plant and animal breeders knew that certain traits were passed on to offspring long before Mendel formulated the quantitative principles of inheritance. One of the best examples of this trend may be found in the history of chemistry, where early scientists discovered qualitative laws of reaction decades before numeric relations were determined. In particular, the history of the theory of acids and bases provides us with useful insights into the discovery of qualitative empirical laws.

By the 17th and 18th Centuries, chemists had made considerable progress in classifying substances on the basis of qualitative properties. During this period, researchers focused on features such as the taste and texture of substances, as well as their interactions with other substances. Thus, they knew that the substance we now call hydrochloric acid had a sour taste, and that it combined with ammonia to form $NH_4Cl$ (though the structure of this compound was not known). Moreover, they knew that sulfuric acid also tasted sour, and that it also combined with ammonia to form $(NH_4)_2SO_4$. From such facts as these, the early chemists defined classes such as *acids*, *alkalis*, and *salts*, and formulated laws involving these terms, such as "acids taste sour" and "acids react with alkalis to form salts". Eventually, they came to view both alkalis and metals as special cases of the more abstract concept of a *base*, and arrived at the more general law that "acids react with bases to form salts". Although some exceptions to these statements were known, chemists found the laws sufficiently general to use in making predictions, as well as in classifying new substances. We shall see that the two processes – defining classes like *acid* and *alkali*, and formulating laws involving these classes – play a central role in the qualitative discovery process.

Table 5. GLAUBER viewed in terms of search concepts.

| | |
|---|---|
| **Initial state:** | a list of facts containing only constants |
| **Goal state:** | a list of laws relating classes, along with definitions of those classes |
| **Intermediate states:** | a list of laws relating some classes, along with definitions of classes; some facts remain |
| **Operators:** | |
| Form-law: | defines a class and substitutes it into facts |
| Determine-quantifier: | specifies existential or universal quantifiers |
| **Heuristics:** | |
| For Form-law: | select the object occurring in the most analogous facts |
| For Determine-quantifier: | quantify universally if the data justify it |
| **Search control:** | Depth-first with no backtracking |

## 3.1. The GLAUBER System

In our efforts to understand the process of scientific discovery, we have also implemented GLAUBER,[3] an AI system that formulates qualitative empirical laws. The program is named after Johann Rudolph Glauber (1604-1670), a 17th Century German chemist who played an important role in developing the theory of acids and bases. Table 5 summarizes GLAUBER in terms of search concepts. The system accepts as input a set of qualitative facts, which are represented in terms of a simple schema. Each fact contains a *predicate* that specifies the type of fact it is, along with one or more attribute-value pairs. For example, the fact that HCl reacts with $NH_3$ to form $NH_4Cl$ would be stored as (reacts inputs {HCl $NH_3$} outputs {$NH_4Cl$}). Here the predicate is *reacts*, the attributes are *inputs* and *outputs*, and the sets {HCl $NH_3$} and {$NH_4Cl$} are the values for these attributes.[4] The knowledge that HCl tastes sour would be stored as (has-quality object {HCl} taste {sour}). In this case the values are enclosed in brackets for consistency with other predicates (such as *reacts*), which may have multiple symbols as values.

[3]The current version of GLAUBER differs from the earlier version described by Langley, Zytkow, Simon, and Bradshaw [6]. Although the state descriptions are very similar in the two systems, both the operators and the search control differ considerably.

[4]GLAUBER knows that the order of symbols contained in a set does not matter, so that (reacts inputs {$NH_3$ HCl} outputs {$NH_4Cl$}) would be considered identical to the above fact. For the convenience of the reader, we will use the contemporary chemical names of substances. From GLAUBER's viewpoint these are simply arbitrary labels with no decodable internal structure.

GLAUBER's goal is to transform these facts into a set of qualitative laws having the same *form* as the original facts, but in which specific substances have been replaced by abstract classes, such as *acid* and *alkali*. In addition, each class must have an associated list of members; for instance, HCl and $H_2SO_4$ would be examples of acids, while NaOH and KOH would be members of the alkali class. Taken together, the qualitative laws relating classes and the extensional definitions of these classes let one predict the original facts, along with other facts that have not yet been observed.

GLAUBER's two operators are concerned with transforming the original data into such laws and classes. The first of these operators, FORM-LAW, inputs a set of facts having the same predicate and at least one common argument; it replaces these with a single law in which some arguments have been replaced by a class name, and defines each of the new classes in terms of their members. For example, given the two facts (reacts inputs {HCl NaOH} outputs {NaCl}) and (reacts inputs {$HNO_3$ NaOH} outputs {$NaNO_3$}), the FORM-LAW operator would define two classes, say x and y, and replace the facts with the law (reacts inputs {x NaOH} outputs {y}). The operator would also note that HCl and $HNO_3$ are members of the newly created class x, while NaCl and $NaNO_3$ are members of the y class. Finally, the FORM-LAW operator iterates through the current set of facts and laws, and replaces occurrences of these substances with their class names. For instance, if the facts (has-quality object {HCl} taste {sour}) and (has-quality object {$HNO_3$} taste {sour}) were known, they would be replaced by the law (has-quality object {x} taste {sour}).

When GLAUBER formulates a new set of laws, the system must decide the appropriate level of generality for each law. To this end, the second operator (DETERMINE-QUANTIFIER) iterates through the set of laws, and determines whether each class mentioned in a law should be existentially or universally quantified. If an existential quantifier is settled on, then the law is interpreted as holding for only a *single* member of the class. If a universal quantifier is selected, the law is interpreted as holding for all members of the class. If a single class is introduced, then this class is universally quantified in the resulting law; in this case, the level of quantification is not an issue, since this is tautologically determined by the manner in which the class was defined. However, if N classes are introduced, then N versions of the law result, each containing one universally quantified class and with the quantifiers for the remaining classes undetermined. For instance, in the above example, two variations on the reaction law would be formulated -- $\forall$x?y (reacts inputs {x NaOH} outputs {y}) and $\forall$y?x (reacts inputs {x NaOH} outputs {y}).[5] The first of these states that all members of class x react with at least one member of the class y; the second states that all members of class y can be formed by at least one member of x in reaction with NaOH. The first quantifier in each law follows from the class definition, but the second quantifier must be determined empirically.

A similar issue arises when the FORM-LAW operator generates additional laws by replacing substances with classes in other facts. In these cases, all of the quantifiers must be tested against observations. For example, the law (has-quality object {x} taste {sour}) might hold for all members of x, or for only a few members of this class. Thus, the DETERMINE-QUANTIFIER operator examines the known facts, and decides on the appropriate quantifier. If more than one class is involved, the possibility of multiple forms of the law must be considered. Thus, if a law were formed by substituting both x and y for members of these classes, GLAUBER might decide on a single law in which both were universally quantified, a single law in which both were existentially quantified, or two laws involving both existential and universal quantifiers.

Once GLAUBER has applied the FORM-LAW and DETERMINE-QUANTIFIER operators, it has a revised set of facts and laws to which these operators can be applied recursively. The FORM-LAW operator

[5]In this paper, expressions of the form $\forall x\ P(x)$ are intended as shorthand for longer expressions of the form $\forall y \mid y \in x\ P(y)$, where x is a class name and y is a member of that class. Expressions of the form $\exists x\ P(x)$ should be interpreted in a similar fashion.

may apply to laws as well as to facts, provided these laws have identical quantifiers. For example, given the two laws $\forall x \exists y$ (reacts inputs {x NaOH} outputs {y}) and $\forall x \exists y$ (reacts inputs {x KOH} outputs {z}), this operator would generate the more abstract law $\forall x \exists w$ (reacts inputs {x u} outputs {w}). In addition, it would define the class u to have the members NaOH and KOH, and define the class w with the classes y and z as subsets. DETERMINE-QUANTIFIER would then proceed to decide on the generality of this law, and the process would be repeated on the revised set of facts and laws. GLAUBER continues this alternation between finding laws and determining their generality until the goal state has been reached — a set of maximally general laws that account for as many of the original facts as possible.

Using its two operators, GLAUBER carries out a *depth-first* search through the space of possible laws and classes. The system's search control does not include backup capability, since its evaluation functions are sufficiently powerful to direct search down acceptable paths. In determining which law to formulate (and thus which classes to define), GLAUBER considers all substances and classes, and selects the symbol that occurs in the largest number of analogous facts. Thus, if two facts having the reacts predicate were found to include NaOH in the inputs slot, then NaOH would receive a score of two, unless it occurred in some other set of facts more often. In the case of laws, GLAUBER uses the total number of facts covered by those laws. GLAUBER indexes its facts and laws in terms of their arguments, so these scores are easily computed for each substance and class. Once this has been done, the system applies the FORM-LAW operator to those facts containing the highest scoring symbol, with the constraint that existentially quantified classes are not considered.

In determining the placement of universal and existential quantifiers, GLAUBER examines the facts (or lower level laws) on which the current law is based. The system generates all of the laws/facts that would be produced by a universal quantifier for a given class, and if enough of these have been observed (or inferred), then the universal quantifier is retained for that class; otherwise an existential quantifier is used. Thus, the system can be viewed as looking ahead one step in order to determine which move is most desirable. A certain percentage of the predicted facts must be observed for GLAUBER to generalize over a class; this percentage is specified by the user. The program interprets missing facts as unobserved; the current system cannot handle disconfirming evidence, such as $\sim \exists$salt (reacts inputs {HCl $HNO_3$} outputs {salt}).

## 3.2. Rediscovering the Concepts of Acids and Alkalis

Now that we have described GLAUBER in the abstract, let us examine its behavior given a particular set of facts as input. These facts are presented at the top of Table 6, and are very similar to facts known by 17th Century chemists before they formulated the theory of acids and bases. As we shall see, GLAUBER arrives at a set of laws and classes very similar to those proposed by the early chemists. The data in the table are intentionally simplified for the sake of clarity. However, we have tested the system on larger sets of data, as well as sets with less regularity.

Given the twelve facts as inputs, GLAUBER begins by examining the symbols used as arguments in the propositions, and determining which of these occur in the greatest number of analogous facts. It notes that the symbols HCl, $HNO_3$, NaOH, and KOH are each arguments of the inputs slot for two facts involving the reacts predicate. Similarly, the symbols sour and bitter each occur as arguments of the taste slot in two has-quality facts. However, the highest scoring symbol is salty, which occurs in four has-quality facts as the value for taste. As a result, these four facts are replaced by the law (has-quality object {salt} taste {salty}), which has the same form as the original propositions, but in which the differing values of the object slot have been replaced by the class name salt. In addition, the four substances NaCl, KCl, $NaNO_3$, and $KNO_3$ are stored as members of the new class.

**Table 6. States generated by GLAUBER in discovering acids and alkalis.**

| | |
|---|---|
| *Initial state S1*: | |
| (reacts inputs {HCl NaOH} outputs {NaCl}) | (has-quality object {NaCl} taste {salty}) |
| (reacts inputs {HCl KOH} outputs {KCl}) | (has-quality object {KCl} taste {salty}) |
| (reacts inputs {$HNO_3$ NaOH} outputs {$NaNO_3$}) | (has-quality object {$NaNO_3$} taste {salty}) |
| (reacts inputs {$HNO_3$ KOH} outputs {$KNO_3$}) | (has-quality object {$KNO_3$} taste {salty}) |
| (has-quality object {HCl} taste {sour}) | (has-quality object {NaOH} taste {bitter}) |
| (has-quality object {$HNO_3$} taste {sour}) | (has-quality object {KOH} taste {bitter}) |
| FIND-LAW and DETERMINE-QUANTIFIER lead to *state S3*: | |
| SALTS: {NaCl, KCl, $NaNO_3$, $KNO_3$} | |
| $\exists$salt (reacts inputs {HCl NaOH} outputs {salt}) | (has-quality object {HCl} taste {sour}) |
| $\exists$salt (reacts inputs {HCl KOH} outputs {salt}) | (has-quality object {$HNO_3$} taste {sour}) |
| $\exists$salt (reacts inputs {$HNO_3$ NaOH} outputs {salt}) | (has-quality object {NaOH} taste {bitter}) |
| $\exists$salt (reacts inputs {$HNO_3$ KOH} outputs {salt}) | (has-quality object {KOH} taste {bitter}) |
| $\forall$salt(has-quality object {salt} taste {salty}) | |
| FIND-LAW and DETERMINE-QUANTIFIER lead to *state S5*: | |
| SALTS: {NaCl, KCl, $NaNO_3$, $KNO_3$} | |
| ACIDS: {HCl, $HNO_3$} | |
| $\forall$acid$\exists$salt (reacts inputs {acid NaOH} outputs {salt}) | (has-quality object {NaOH} taste {bitter}) |
| $\forall$acid$\exists$salt (reacts inputs {acid KOH} outputs {salt}) | (has-quality object {KOH} taste {bitter}) |
| $\forall$salt(has-quality object {salt} taste {salty}) | |
| $\forall$acid (has-quality object {acid} taste {sour}) | |
| FIND-LAW and DETERMINE-QUANTIFIER lead to final *state S7*: | |
| SALTS: {NaCl, KCl, $NaNO_3$, $KNO_3$} | |
| ACIDS: {HCl, $HNO_3$} | |
| ALKALIS: {NaOH, KOH} | |
| $\forall$alkali$\forall$acid$\exists$salt (reacts inputs {acid alkali} outputs {salt}) | |
| $\forall$salt(has-quality object {salt} taste {salty}) | |
| $\forall$acid (has-quality object {acid} taste {sour}) | |
| $\forall$alkali (has-quality object {alkali} taste {bitter}) | |

In addition to proposing this law, the FORM-LAW operator generates four additional laws by substituting the symbol salt for members of this class into other facts. Thus, the facts (reacts inputs {HCl NaOH} outputs {NaCl}) and (reacts inputs {HCl KOH} outputs {KCl}) are replaced by the laws (reacts inputs {HCl NaOH} outputs {salt}) and (reacts inputs {HCl KOH} outputs {salt}). Similarly, the facts (reacts inputs {$HNO_3$ NaOH} outputs {$NaNO_3$}) and (reacts inputs {$HNO_3$ KOH} outputs {$KNO_3$}) are replaced by (reacts inputs {$HNO_3$ NaOH} outputs {salt}) and (reacts inputs {$HNO_3$ KOH} outputs {salt}). Although the first of these laws is guaranteed to be universally quantified by the manner in which the salt class was defined, the generality of the other laws must be determined empirically. For example, if the law (reacts inputs {HCl NaOH} outputs {salt}) were universally quantified over the class of salts, then four facts would be predicted. Since only one of these predictions has been observed, GLAUBER employs an existential quantifier rather than a universal one. The same decision is made for the other laws formed by substitution, leading to the laws and facts shown in the second section of the table.

Given this new state of the world, GLAUBER again determines which of the known symbols occur in the most analogous facts. In this case, the set of alternatives is slightly different from that on the earlier cycle, since the class name **salt** has replaced the individual members of that class. Given the current set of facts and laws, six symbols tie for the honors – NaOH, KOH, HCl, $HNO_3$, sour, and bitter. For example, the first of these occurs in the laws ∃salt (reacts inputs {HCl NaOH} outputs {salt}) and ∃salt (reacts inputs {$HNO_3$ NaOH} outputs {salt}), while the second occurs in the laws ∃salt (reacts inputs {HCl KOH} outputs {salt}) and ∃salt (reacts inputs {$HNO_3$ KOH} outputs {salt}). The **salt** symbol actually occurs in all four of these laws, but the class is not considered, since it is existentially quantified in these laws. Since all of the viable options involve two laws (each based on one fact apiece), GLAUBER selects one of them at random. Let us follow the course events take when the system chooses the pair of facts involving the symbol NaOH.

Based on these facts, the FORM-LAW operator generates the law (reacts inputs {acid NaOH} outputs {salt}), and defines the new class **acid** as containing the elements HCl and $HNO_3$. Two additional laws result from substitution – (reacts inputs {acid KOH} outputs {salt}) and (has-quality object {acid} taste {sour}) – each replacing two directly observed facts. After substitution, GLAUBER has four laws and two facts in memory. However, the system must still determine the generality of these laws. The DETERMINE-QUANTIFIER operator proceeds to consider the predictions made by each law when universally quantified over the new class of acids. Since all of the predicted facts have been observed, the universal quantifier is retained for each of the new laws, giving the set of facts and laws shown in the third section of the table.

At this point, only five symbols remain to be considered – NaOH, KOH, bitter, and the classes **salt** and **acid**. The first two occur only in single laws, while the third occurs in two analogous facts. The class name **salt** appears in two analogous laws, but is ignored due to its existential quantifier. However, the class name **acid** occurs in two analogous laws that are based on two facts apiece, giving **acid** a score of four. As a result, the two laws are passed to the FORM-LAW operator and a higher level law – (reacts inputs {acid alkali} outputs {salt}) – is formed on this basis. In addition, the class **alkali** is defined as having the members NaOH and KOH. A second law – (has-quality object {alkali} taste {bitter}) – is formed by substitution, and both laws are universally quantified over the new class, the first by definition and the second empirically. At this point, GLAUBER has reached its goal of specifying a maximally general set of laws that summarize the original data. The final laws are shown in the fourth section of Table 6, and are very similar to those proposed by the early chemists. When GLAUBER is given reactions involving metals as well as alkalis, it defines the broader class of bases (containing both metals and alkalis as members), and arrives at the central tenet that acids combine with bases to form salts.

## 3.3. Comments on GLAUBER

In its present form, GLAUBER has some important limitations, which should be remedied in future versions of the system. The first difficulty relates to the system's evaluation function for directing search through the space of classes and laws. The current version iterates through the set of known symbols, and selects that symbol which occurs in the greatest number of analogous facts. This leads GLAUBER to prefer large classes to small ones, which in turn leads to laws with greater generality, in the sense that they cover more of the observed facts. However, recall that once GLAUBER defines a new class on the basis of some law, it then creates additional laws by substituting the class for its members in other facts. This suggests a broader definition of generality, including all facts predicted by any law involving the new class. This analysis leads to two methods for preferring one class over another. The most obvious approach involves computing the percentage of predictions that are actually borne out by observations; we shall call this the *predictive power* of a class and its associated laws. The second method involves computing the total number of facts predicted by a class and its related laws; we shall call this the *predictive potential* of the class.

Obviously, a set of laws that predicts a few observations but predicts many unobserved ones is undesirable; this suggests that predictive power should be used to weed out grossly unacceptable classes. However, given roughly equal scores on this dimension, sets of laws with greater predictive potential should be preferred, since these lead to many predictions which, if satisfied, will lead to an increase in predictive power. One difficulty in implementing this scheme is that GLAUBER would have to generate the potential classes and their associated laws in order to determine their predictive power and potential. Moreover, it would have to consider whether these laws should be existentially or universally quantified in order to compute their scores. In other words, the system would have to apply the FIND-LAW operator in all possible ways, and then apply the DETERMINE-QUANTIFIER operator in all possible ways in order to determine the best path to follow. Since this is equivalent to doing a two-step lookahead in the search tree, it would involve considerably more computation time than the current simple strategy. The details of this scheme remain to be elaborated, but the basic idea of defining classes that account for the most data seems a plausible approach.

A second limitation involves the possibility of alternate divisions of substances into classes. In some cases, two or more branches in the search tree may lead to equally (or near-equally) good descriptions of the data. These competing paths may ultimately lead to the same state, or they may lead to completely different organizations of knowledge.. In the latter case, one would like the system to discover both frameworks. However, since the current version of GLAUBER carries out a depth-first search without backup, it must select one of the paths at random, thus ignoring what may be an equally useful summary of the data. Future versions of the system should be able to consider multiple alternatives while still using evaluation functions to keep search to a minimum.

In order to understand the last of GLAUBER's limitations, we must review some related work on machine learning. Wolff [7] has explored an approach to grammar learning that incorporates methods very similar to those used in GLAUBER. Wolff's system begins with a sequence of letters, and based on common sequences of symbols, defines *chunks* in terms of these sequences. For example, given the sequence "thedogchasedthecatthecatchasedthedog...", the program defines chunks like **the**, **dog**, **cat**, and **chased**. Whenever a chunk is created, the component symbols are replaced by the symbol for that chunk. In this case, the sequence "the-dog-chased-the-cat-the-cat-chased-the-dog" would result. In addition, when a number of different symbols (letters or chunks) are found to precede or follow a common symbol, a disjunctive class is defined in terms of the first set. For instance, in the above sequence we find the subsequences "the-dog-chased" and "the-cat-chased)". Based on this regularity, Wolff's program would define the disjunctive class **noun** = {**dog**, **cat**}. The symbol for this new class is then substituted into the letter sequence for the member symbols. In this case, the sequence "the-noun-chased-the-noun-the-noun-chased-the-noun" would be generated. These two basic methods are applied recursively, so that chunks can be defined in terms of disjunctive classes, and vice versa. Thus, given the last sequence, the chunk **sentence** = **the-noun-chased-the-noun** would be defined, giving the final sequence "sentence-sentence".

From this description we see that Wolff's learning system employs two operators -- one for forming disjunctive classes such as **noun**, and another for defining chunks or *conjunctive* classes, such as **dog**. The first of these is identical to GLAUBER's operator for forming disjunctive classes like acid and **alkali**.[6] The main difference between the two systems lies in the *heuristics* for forming such disjuncts. Wolff employs adjacency criteria well-suited to the language acquisition domain, while GLAUBER uses the notion of shared

[6]Rather we should say that GLAUBER's operator is identical to Wolff's operator, since Wolff's work preceded our own by many years. Although the original version of GLAUBER was developed independently of Wolff's approach, the current system borrows considerably from his results in the domain of grammar learning.

arguments, which are more appropriate for relational domains. In contrast, the second operator in Wolff's method has no analog in GLAUBER's repertoire, and this suggests a gap in our discovery system's capabilities. Upon reflection, one would like GLAUBER to note recurring relations between *conjunctions* of facts, as well as those involving isolated propositions. Let us consider an example from the domain of genetics that requires this form of reasoning. Suppose the system observed (as did Mendel) that when certain green garden peas were self-fertilized, they produced only green offspring, but that when other green peas were self-fertilized, they produced both green and yellow offspring. In this case, we would like GLAUBER to divide the green peas into two classes based not on their own directly observable features (since these are identical), but based on the features of their offspring. Thus, in looking for patterns, the system would have to examine not only single facts, but pairs of facts, triples of facts, and so forth. Such a strategy, though much more expensive than the current one, would enable the program to note that some green peas have only green offspring, while others have mixed offspring, and to classify them on this basis. This would be equivalent to defining *chunks* based on co-occurring facts, and can be viewed as a relational version of the chunking method used in Wolff's system.

We should also consider briefly some other discovery systems with similar concerns. First, Michalski and Stepp [8] have studied the task of conceptual clustering, in which one forms a hierarchical taxonomy for classifying objects. Since GLAUBER also divides objects into classes, it can be viewed as carrying out a form of conceptual clustering, even though its methods differ significantly from those used by Michalski and Stepp. GLAUBER also bears some resemblance to Brown's [9] discovery system, which also generated abstract laws covering a set of facts. However, this early system's search methods also differed considerably from those in GLAUBER, and it did not define new classes in the process of stating laws. Finally, we should mention some recent work by Emde, Habel, and Rollinger [10] that also involves the discovery of qualitative laws. In this case, the focus is on determining whether predicates obey certain relations, such as transitivity or inversivity. Although this approach leads to laws very similar to those found by Brown, the model-driven discovery method contrasts with the data-driven approach used in the other systems. To summarize, we find that GLAUBER bears some relation to other systems for qualitative discovery, but is most similar to Wolff's grammar learning system in both spirit and method.

## 4. Determining the Components of Substances

We have already seen that early chemists were concerned with both qualitative and quantitative descriptions of chemical reactions. However, another one of their primary goals was to determine the *components* of various substances, and information about chemical reactions proved quite useful in this regard. The goal of determining such components became an important aspect of the atomic theory, in that it postulated primitive building blocks for the observed substances, even though no stance was taken on whether these building blocks were particulate or continuous in nature. Thus, the formulation of componential models embodied a simple form of explanation that is clearly distinct from the descriptive summaries generated by BACON and GLAUBER.

During the 18th Century, chemists developed models of many substances, but they devoted considerable attention to explaining combustion and related phenomena. As a result, two different componential models were eventually proposed to account for this process. The first assumed that combustion involved the decomposition of two substances, and was known as the theory of *phlogiston*. The second assumed that combustion involved the combination of two substances, and was called the *oxygen* theory. Although the phlogiston theory was eventually rejected in favor of its competitor, we will see that it provided a plausible account of the known reactions, and was well-respected for decades. This suggests an important constraint on computational models of scientific discovery: such models should be able to arrive at plausible laws or models even if they were ultimately rejected in favor of others. This makes the area of combustion reactions an ideal test for systems concerned with formulating componential models, since we know two models that can usefully account for the observations.

Table 7. STAHL viewed in terms of search concepts.

**Initial state:** a list of reactions relating substances

**Goal state:** the components of each compound substance

**Intermediate states:** components of some substances, modified reactions

**Operators\Heuristics:**
- **Infer-composition:** decides on the components of a substance
- **Reduce:** cancels substances occurring on both sides of a reaction
- **Substitute:** replaces a substance with its components in a reaction
- **Identify-components:** identifies two components as the same
- **Identify-compounds:** identifies two compounds as the same

**Search control:** Depth-first search with no backtracking

## 4.1. The STAHL System

Our interest in componential models has led us to construct a third AI system that infers such models from a set of known reactions. The program is named STAHL, after G. E. Stahl (1660 - 1734), one of the principal formulators of the phlogiston theory. Like GLAUBER, this program accepts qualitative information as input, and generates qualitative statements as output. However, since STAHL's conclusions relate to the internal structure of substances, they can be viewed as simple explanations rather than descriptive summaries. The system's initial state consists of a set of reactions, represented in the same schema-like format used by GLAUBER. For instance, the reaction of hydrogen and oxygen to form water would be represented as (reacts inputs {hydrogen oxygen} outputs {water}). STAHL's goal is to determine the components of all non-elemental substances involved in the given reactions. This information is represented in the same formalism as the initial reactions. Thus, the conclusion that water is composed of hydrogen and oxygen would be stated as (components of {water} are {hydrogen oxygen}). Intermediate states consist of inferences about the components of some substances, along with transformed versions of the initial reactions.

STAHL incorporates four operators for moving through the space of possible componential models. These operators are closely linked to the heuristics that propose them, so they are best discussed together. The most basic of these operator/heuristics deals with simple synthesis and decomposition reactions, and lets the system infer unambiguously the components of a compound. It can be stated:

```
INFER-COMPOSITION
If A and B react to form C,
  or if C decomposes into A and B,
then infer that C is composed of A and B.
```

An obvious example of this rule's use involves determining the components of water. Given the information that hydrogen reacts with oxygen to form water, STAHL would infer that the latter substance is composed of the first two. Note that STAHL does not draw any conclusions about the *amount* of hydrogen and oxygen contributing to water, but only that they contribute something. Of course, the INFER-COMPOSITION rule is not limited to reactions involving pairs of elements, but can also deal with cases in which three or more substances unite to form a single compound.

If all chemical reactions were of the form shown above, STAHL's task would be simple indeed. However, more complex reactions are common in chemistry, so STAHL includes additional operators for dealing with these more complex situations. The purpose of these operators is to transform complex reactions into simpler forms, so they can eventually be matched by the INFER-COMPOSITION rule shown above. One such operator is responsible for "canceling" out substances occurring on both sides of a reaction; the reduction heuristic which proposes this operator can be paraphrased:

```
REDUCE
If A occurs on both sides of a reaction,
then remove A from the reaction.
```

This heuristic leads directly to a simplified version of a reaction. For instance, if STAHL is told that "A, B, and C react to form D and C", the REDUCE rule would apply, giving the simplified reaction "A and B react to form D". This revised relation would then be used by the INFER-COMPOSITION rule to infer that D is composed of A and B. One can imagine cases in which this approach would lead to errors, as when different amounts of a substance are observed before and after a reaction. However, one can also imagine more conservative versions of the heuristic that require equal amounts of the canceled substance to occur on each side.

STAHL incorporates a third operator that initially leads to more complex statements of reactions, but may make it possible for the REDUCE rule to apply. The heuristic for proposing this operator draws on information about the components of a substance that have been inferred earlier; it can be stated:

```
SUBSTITUTE
If A occurs in a reaction,
  and A is composed of B and C,
then replace A with B and C.
```

For instance, the system may know that X is composed of Y and Z, and that "X reacts with W to form V and Z". In this case, the SUBSTITUTE rule would rewrite the second relation as "Y, Z, and W react to form V and Z". Given this formulation, the REDUCE rule would lead to "Y and W react to form V", and the INFER-COMPOSITION rule would lead to the conclusion that V is composed of Y and W. As before, the SUBSTITUTE rule is not restricted to substances composed of two elements, but works equally well for more complex structures.

A final operator is responsible for postulating that two substances that were originally thought to be different are in fact identical. Two separate heuristics propose when to apply this operator; the first of these rules may be stated:

```
IDENTIFY-COMPONENTS
If A is composed of B and C,
  and A is composed of B and D,
then identify C with D.
```

This heuristic matches when STAHL learns that a compound can be decomposed in two different ways, but the decompositions differ by a only single substance. The second heuristic is very similar, except that it applies when two apparently different compounds are found to have the same components. It can be paraphrased:

**IDENTIFY-COMPOUND**
**If A is composed of C and D,**
  **and B is composed of C and D,**
**then identify A with B.**

The history of chemistry abounds with cases in which a new substance was discovered in two different contexts, was originally thought to be two distinct substances, and was eventually combined into a single concept. We will see an example of such identification shortly.

STAHL can be viewed as carrying out a depth-first search through the space of componential models, relying entirely on its heuristics to select the appropriate path. In general, these heuristics are sufficiently powerful that the system need never backtrack, though we will discuss some situations later where backtracking is required. In some cases, more than one heuristic (or more than one instantiation of the same heuristic) can be applied to the current state. When this occurs, one instantiation is selected at random. In the runs we have carried out, this random selection has not significantly affected the final inferences made by the system, though it does affect the intermediate states that are generated.

Table 8. Inferring the composition of lime and magnesia.

Initial state S1:
(reacts inputs {lime} outputs {quick-lime fixed-air})
(reacts inputs {quick-lime magnesia} outputs {lime calcined-magnesia}).

INFER-COMPOSITION leads to state S2:
(components of {lime} are {quick-lime fixed-air})
(reacts inputs {quick-lime magnesia} outputs {lime calcined-magnesia}).

SUBSTITUTE leads to state S3:
(components of {lime} are {quick-lime fixed-air})
(reacts inputs {quick-lime magnesia} outputs {quick-lime fixed-air calcined-magnesia})

REDUCE leads to state S4:
(components of {lime} are {quick-lime fixed-air})
(reacts inputs {magnesia} outputs {fixed-air calcined-magnesia})

INFER-COMPOSITION leads to final state S5:
(components of {lime} are {quick-lime fixed-air})
(components of {magnesia} are {fixed-air calcined-magnesia})

One of STAHL's interesting features is the manner in which its heuristics interact. Note that the substitution rule requires knowledge of a substance's composition, so that some inferences about composition must be made before it can be used. However, we have also seen that complex reactions must be rewritten by the reduction and substitution rules before some composition inferences can be made. This interdependence leads to a "bootstrapping" effect, in which inferences made by one of the rules enable further inferences to be made, these allow additional inferences, and so forth, until as many conclusions as possible have been drawn. This process generally begins with one or more simple reactions, but after this the particular path taken depends on the data available to the system.

Let us consider STAHL's heuristics in operation on the relatively simple task of inferring the composition of lime and magnesia. In order to formulate models of these two substances, the system requires two initial reactions: (reacts inputs {lime} outputs {quick-lime fixed-air}) and (reacts inputs {quick-lime magnesia} outputs {lime calcined-magnesia}). Given this information, the INFER-COMPOSITION rule applies first, leading to the inference that lime ($CaCO_3$) is composed of quick-lime ($CaO$) and fixed-air ($CO_2$). This result enables the SUBSTITUTION heuristic to match, leading to a temporarily more complex version of the second reaction, (reacts inputs {quick-lime magnesia} outputs {quick-lime fixed-air calcined-magnesia}). However, since the substance quick-lime occurs in both sides of the modified reaction, the REDUCTION rule applies, transforming it into the simpler form (reacts inputs {magnesia} outputs {fixed-air calcined-magnesia}). Finally, this reduced form allows the INFER-COMPOSITION rule to infer that magnesia is composed of the substances fixed-air ($CO_2$) and calcined-magnesia ($MgO$). At this point, since no more of its heuristics seem applicable, STAHL concludes that it has formulated as many componential models as the data allow, and halts its operation. The system's behavior on this example is summarized in Table 8. Now that we have presented an overview of STAHL's inference methods, let us examine their application to a historically more interesting example — discovering the phlogiston theory.

## 4.2. Discovering the Phlogiston Theory

The theory of phlogiston originated early in the 18th Century, and after undergoing several transformations, was widely accepted until the 1780's. This theory adopted the ancient view that fire, heat, and light are different manifestations of a common principle that leaves a body during combustion. Therefore, any reaction involving combustion was viewed as a decomposition; for instance, burning coal was interpreted as decomposing into the matter of fire (another term for phlogiston) and ash.[7] Early phlogistians were not able to isolate phlogiston, but the disengagement of fire during combustion seemed to be a good observational reason for admitting the disengagement of a substance from the burning body. Later, as the notion of phlogiston proved useful in explaining many additional reactions, the existence of this substance was supported by a substantial body of evidence.

After they began to study combustion within closed vessels, chemists realized that air was necessary for combustion to occur. However, they did not assume that air changed its chemical identity during this process. Rather, they decided that air played an auxiliary role, similar to that played by water in reactions involving acids, alkalis, and salts. Thus, even starting with empirically more complete descriptions of combustion, such as "in the presence of air, carbon burns to release phlogiston and to form ash", they employed the reduction heuristic to remove air and simplify the relation. Given such data, STAHL makes similar "errors" in reasoning, so that it provides a simple explanation of the process by which chemists developed phlogiston-based models of combustion reactions. Such confusions are common in the history of chemistry, and a similar error led the followers of Lavoisier (around 1810) to believe that sodium was a compound of soda and hydrogen.

Let us examine the path taken by STAHL in arriving at one version of the phlogiston theory. We present the system with two facts: (reacts inputs {coal air} outputs {matter-of-fire ash air}) and (reacts inputs {calx-of-iron coal air} outputs {iron ash air}).[8] One may question the exact representation of these facts, but clearly something very much like them was believed during the period in which the phlogiston theory was developed. Given this information, STAHL immediately applies its REDUCE operator to the first fact, giving the revised reaction (reacts inputs {coal} outputs {matter-of-fire ash}). The system then applies the same

[7]Several decades later, in the second half of the 18th Century, fixed air (carbon dioxide) was discovered, and recognized as the product of burning coal in place of ash.

[8]Calx of iron was the current name for iron oxide; we have used the original terminology because the modern term is based on the oxygen theory developed by Lavoisier.

operator to the second fact, giving the reduced reaction (reacts inputs {calx-of-iron coal} outputs {iron ash}). After this, the first of these revisions, combined with the INFER-COMPOSITION rule, leads to the inference that coal is composed of matter-of-fire (or phlogiston) and ash, which was one tenet of the early phlogiston theory. Having arrived at this conclusion, STAHL applies the SUBSTITUTE rule, generating the expanded relation (reacts inputs {calx-of-iron ash matter-of-fire} outputs {iron ash}). At this point, the REDUCE rule is used to remove ash from both sides of the equation, giving (reacts inputs {calx-of-iron matter-of-fire} outputs {iron}). Finally, the INFER-COMPOSITION operator leads STAHL to infer that iron is a compound composed of calx-of-iron and the matter of fire. Table 9 summarizes the states visited by the system in arriving at these conclusions, along with the operators used to generate them.

Table 9. STAHL's steps in formulating the phlogiston model.

Initial state S1:
(reacts inputs {coal air} outputs {matter-of-fire ash air})
(reacts inputs {calx-of-iron coal air} outputs {iron ash air})

REDUCE leads to state S2:
(reacts inputs {coal} outputs {matter-of-fire ash})
(reacts inputs {calx-of-iron coal air} outputs {iron ash air})

REDUCE leads to state S3:
(reacts inputs {coal} outputs {matter-of-fire ash})
(reacts inputs {calx-of-iron coal} outputs {iron ash})

INFER-COMPOSITION leads to state S4:
(components of {coal} are {matter-of-fire ash})
(reacts inputs {calx-of-iron coal} outputs {iron ash})

SUBSTITUTE leads to state S5:
(components of {coal} are {matter-of-fire ash})
(reacts inputs {calx-of-iron matter-of-fire ash} outputs {iron ash})

REDUCE leads to state S6:
(components of {coal} are {matter-of-fire ash})
(reacts inputs {calx-of-iron matter-of-fire} outputs {iron})

INFER-COMPOSITION leads to final state S7:
(components of {coal} are {matter-of-fire ash})
(components of {iron} are {calx-of-iron matter-of-fire})

Let us now consider how STAHL employs its identification heuristics with respect to the phlogiston theory. Suppose we give the system the following additional data: (reacts inputs {iron sulfuric-acid water} outputs {vitriol-of-iron inflammable-air water}) and (reacts inputs {calx-of-iron sulfuric-acid water} outputs {vitriol-of-iron water}).[9] Given these facts, STAHL removes the water from both reactions using the REDUCE operator. This sufficiently simplifies the second reaction so that it can apply the INFER-COMPOSITION rule, inferring that vitriol-of-iron is composed of calx-of-iron and sulfuric-acid. This fact is

[9]The formula for vitriol of iron is $FeSO_4$, while the modern name for inflammable air is hydrogen.

substituted into the first reaction, giving (reacts inputs {iron sulfuric-acid} outputs {calx-of-iron sulfuric-acid inflammable-air}). After using the REDUCE operator to eliminate sulfuric-acid from both sides of this expression, STAHL infers that iron consists of calx-of-iron and inflammable air. However, the system knows from the other reactions described earlier that iron can also be decomposed into calx-of-iron and phlogiston. Using the first of its identification heuristics (IDENTIFY-COMPONENTS), the system infers that inflammable-air and phlogiston are identical. Both the reasoning and conclusions of STAHL in this example are very similar to those of Cavendish and other phlogiston theorists during the 1760's.

### 4.3. Comments on STAHL

Earlier we mentioned one case in which STAHL's heuristics might lead to erroneous inferences, but did not pursue the matter. In fact, there are a number of ways in which the system's heuristics can lead it astray, and we are currently extending the system to deal with these cases. One situation involves the notion of infinitely recursing componential models. For instance, given certain reactions involving mercury, calx-of-mercury, and oxygen,[10] STAHL eventually makes two inferences: (components of {mercury} are {calx-of-mercury phlogiston}) and (components of {calx-of-mercury} are {mercury oxygen}). Taken together, these two inferences imply that mercury is composed of itself, and this seems an undesirable characteristic for an explanatory model.

Ultimately, such infinite recursions must be due to the faulty description of one or more reactions. Given trace information about which heuristics proposed which inferences, an extended version of STAHL should be able to track down the responsible reaction and call it into question. Historically, chemists introduced conceptual distinctions to explain such inconsistencies. For instance, to avoid the difficulty mentioned above, they formulated the concept *calx-of-mercury-proper* as distinct from calx-of-mercury. In some sense, this is similar to BACON's introduction of new intrinsic properties when it encounters a situation in which its numeric methods fail to apply. As with BACON, such concepts may appear tautological when first introduced, but become respectable to the extent that they prove useful in dealing with other situations besides the one leading to their introduction.

STAHL's heuristics can lead to other forms of inconsistency as well. For instance, the system may infer that A consists of B and C, and later infer that A also consists of B, C, and D. Alternately, the program may reduce a reaction to the form (reacts inputs {X} outputs { }), which contains inputs but no outputs. In both cases, an extended version of STAHL should be able to trace back through its chain of inferences to determine the source of the problem, and either reject the offending observation or restate it using a new concept. This process can be viewed as a form of backtracking through the search space, though not in any simple sense. It is better described as rejecting the current state and moving sideways through the problem space to another state at approximately the same depth. In any case, such backtracking methods would make STAHL a more robust discovery system in spite of its occasionally misleading heuristics. Moreover, it would improve the system's status as a historical model, since such reformulations occurred many times in the early days of chemistry.

## 5. Formulating Structural Models

As an area of science matures, researchers progress from descriptions to explanations. Although the dividing line between these forms of understanding is fuzzy, some examples clearly lie at the explanatory end of the spectrum. For instance, the kinetic theory of heat provides an explanation of both Black's law and the ideal gas law. A simpler example, though no less impressive at the time it was proposed, is Dalton's atomic

[10]Later versions of the phlogiston theory actually included oxygen as an element, but retained phlogiston as their central feature.

theory. Both examples involve some form of structural model, in which macroscopic phenomena are described in terms of their inferred components. Although this is not the only form of scientific explanation, the notion of structural models seems significant enough to explore in some detail. Let us review the history of the atomic theory as a prelude to our computational analysis of this aspect of discovery.

We have seen that a portion of the atomic hypothesis was implicit in componential models such as the phlogiston theory, but the full version of the atomic model was first proposed by John Dalton in 1808. In his attempt to explain the law of multiple proportions, Dalton assumed that substances were composed of particles called atoms, and focused on the *numbers* of particles making up each substance. Following his lead, chemists adopted the design of such atomic models as one of their central concerns. Dalton employed his *rule of greatest simplicity* to apply the atomic theory to specific cases, and though this heuristic worked in many cases, it led him to incorrect conclusions in others. For instance, it led him to conclude that water was composed of a single hydrogen atom and a single oxygen atom. In contrast, Avogadro (1811) employed Gay-Lussac's data on combining volumes, along with the assumption that equal volumes of gases contained equal numbers of particles. Using this information, he inferred diatomic models for hydrogen and oxygen and a different structure for water. Although we accept Avogadro's hypothesis today, it was rejected by his contemporaries, since they believed that different atoms of the same element would repel, rather than attract, each other; hence, that diatomic molecules of elements were impossible. This is another case in which two hypotheses provided plausible accounts of phenomena, making the area an ideal one for testing a discovery system concerned with formulating structural models.

Table 10. DALTON viewed in terms of search concepts.

**Initial state:** a list of reactions and the component of the substances involved

**Goal state:** a model of each reaction, specifying the number of molecules
and the number of particles in each compound

**Intermediate states:** partials models of some reactions

**Operators:**
- **Determine-molecules:** specifies the number of times a compound occurs in a reaction
- **Determine-atoms:** specifies the number of atoms of a given type in a molecule
- **Conserve-particles:** determines remaining numbers based on conservation principle

**Heuristics:**
- For Determine-molecules: consider only multiples of the combining volumes
- For Determine-atoms: select simpler models first

**Search control:** Depth-first search with backtracking

### 5.1. The DALTON System

Our interest in structural models led us to construct a fourth discovery system concerned with this issue. Since John Dalton was one of the earliest proponents of such models, we have named the system DALTON. The system accepts a set of reactions as input, along with information about the components of the substances involved in these reactions. For instance, DALTON would be told that hydrogen reacts with oxygen to form water, and that hydrogen reacts with nitrogen to form ammonia. Along with this information, the system would be told that water has hydrogen and oxygen as its components, while ammonia has hydrogen and nitrogen as its components.[11] Finally, it would be informed that hydrogen, oxygen, and nitrogen are elements, implying that they have no components other than themselves.

DALTON knows that two quantities are important in a reaction — these are the number of *molecules* of each substance that take part (in the simplest form of the reaction), and the number of *particles* (atoms) of each type in a given molecule.[12] The system's goal is to devise a model for each reaction that specifies the number of molecules and particles for each of the substances involved. Given this goal, the reactions from which DALTON begins its search are best viewed as very abstract models in which these numbers have not yet been specified. In its search through the space of models, the program generates intermediate states in which some amounts have been specified but others have not. Table 10 summarizes the program in terms of search concepts.

The system incorporates three operators for instantiating these models, and thus moving through the problem space. The first operator inputs a reaction in which the number of molecules for a particular substance is unknown, and outputs a revised reaction in which this number is specified. For instance, this routine must hypothesize the number of oxygen molecules involved in the water reaction. A second operator is responsible for specifying the number of times a given component occurs in a particular substance. For example, given the information that oxygen is one of the components of sulfuric acid, this operator would hypothesize the number of oxygen atoms in the acid. A final operator also determines the number of atoms in a substance, but in a much more efficient manner. This routine assumes that for each element taking part in a reaction, the total number of particles is conserved. The operator is given the number of molecules on both sides of a reaction, along with the number of particles on one side of that reaction. From this information, it determines whether the conservation assumption can be satisfied, and if so, it specifies the number of particles on the other side of the reaction necessary to balance the equations. If conservation cannot be satisfied under the existing assumptions, it returns this information instead.

Using these three operators, DALTON carries out a *depth-first* search through the space of possible models. The system focuses on one reaction at a time, first determining the number of molecules and then the number of particles in each molecule. Simpler models are considered before more complex ones. Thus, a model involving one molecule for some substance would be proposed before one specifying two or three molecules. Similarly, models incorporating one occurrence of an element (monatomic models) would be considered before models involving two occurrences of that element (diatomic models). The conservation assumption is employed as soon as the model for a reaction is sufficiently constrained for it to be used. Since some partial models cannot be instantiated in any way that will satisfy the conservation constraint, DALTON must be able to backtrack and consider other paths to a complete model.

One additional constraint makes the process of constructing models challenging. Consistency requires that the model for a substance be the same for all reactions in which it occurs. For example, if hydrogen is assumed to be monatomic for the water reaction, it must also be monatomic in the ammonia reaction. In general, this assumption will simplify the search process, since models completed earlier will constrain those dealt with at later points. However, it is possible that the model for a substance results in a conservation-consistent model for one reaction, but leads to difficulties for another reaction. In such cases, DALTON must revise its earlier model in order to construct a consistent explanation for both reactions. This involves a form

[11] Thus, DALTON accepts as input the type of information that STAHL generates as output, suggesting that these systems could be could be easily linked together. We will discuss this possibility in a later section.

[12] This means that the DALTON program begins with a better notion of the true situation that did its namesake, since John Dalton did not make the distinction between atoms and molecules.

of backtracking, though not the simple form discussed above, since some existing models may be retained. We will see an example of this backup method shortly.

## 5.2. A Monatomic Model of the Water Reaction

Now that we have examined DALTON's problem space and search control in the abstract, let us consider their use in an example. Suppose the system is asked to construct a model of the water reaction, given the information that water is composed of hydrogen and oxygen, and that hydrogen and oxygen are primitive elements (and thus composed of themselves). In this case, the program must determine the number of hydrogen, oxygen, and water molecules, and the number of atoms of each type in the various molecules. As we have seen, DALTON begins with a very abstract model in which no commitments are made, and successively refines this model as it proceeds. Let us examine what happens at each stage in the search through the space of models.

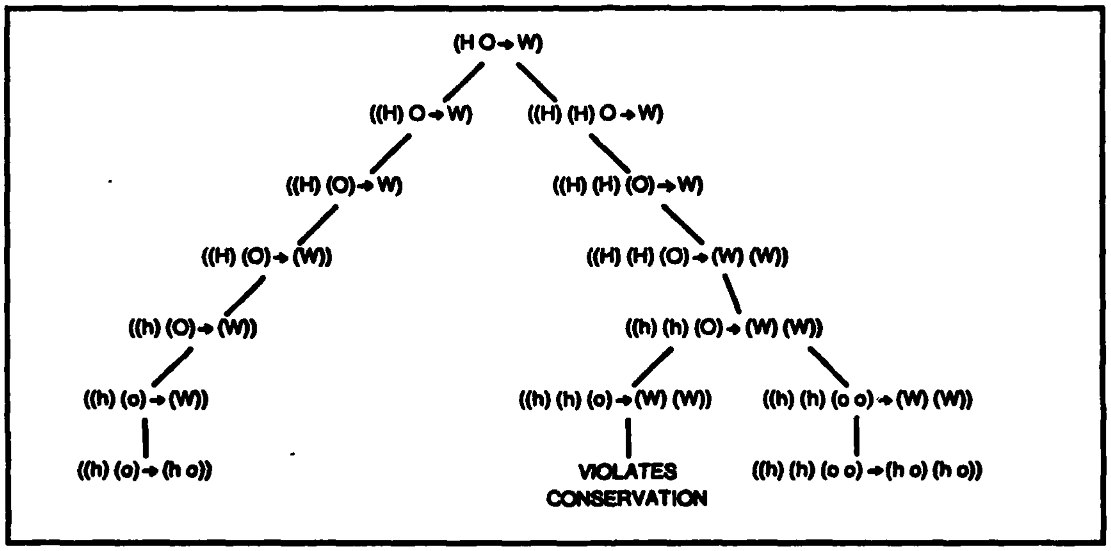


Figure 3. DALTON's search for a model of the water reaction.

Starting with an abstract model of the form (H O → W), the program first considers the number of hydrogen molecules involved. Lacking any theoretical bias, the system chooses the simplest hypothesis and assumes a single hydrogen molecule is required. If this choice later causes difficulty, the model-builder can back up and try another path. Similar initial choices are made for oxygen and water, so that the partially specified model includes one molecule each. This is represented by the proposition ((H) (O) → (W)), in which each molecule is enclosed in parentheses.

Now DALTON must determine the internal structure of each type of molecule, and it decides to assume initially that both hydrogen and oxygen consist of a single atom (say h and o), giving the model ((h) (o) → (W)). At this point, the program invokes its conservation-based operator. This routine checks to see if the model can be finalized in such a way that conservation is obeyed. If this is possible, DALTON outputs the completed model and halts, but if the conservation principle cannot be satisfied, the system backs up and considers other possibilities. In this case, the conservation operator tells DALTON that the water molecule must be composed of one h particle and one o particle, and that the final model must have the form ((h) (o) → (h o)). This model is equivalent to the one originally formulated by the human chemist, John Dalton. Figure 3 presents some of the paths available in the space of molecular models. In arriving at the monatomic model just described, DALTON takes the left path, and since this leads to an acceptable solution, no backtracking is required.

## 5.3. A Diatomic Model of the Water Reaction

As we have seen, DALTON's basic strategy is to carry out a depth-first search through the space of models, ordering the search so that simple models are considered first. However, when enough of the model has been specified, a theory-driven heuristic (implementing the conservation assumption) takes over and finalizes the model. DALTON can also employ theory-driven methods at other stages in its search process, and these methods can alter the system's behavior in significant ways. Thus, in the above run, the system had no theoretical biases other than a belief in conservation of atoms and a desire to construct as simple a model as possible. However, if we give DALTON some additional information about the water reaction, its behavior changes significantly. Avogadro was aware of Guy-Lussac's results, and believed that the combining volumes he observed were related to the number of molecules involved in the reaction. To model this knowledge, we can add the heuristic:

```
INFER-MULTIPLES
If you want to know the number of molecules of X that are involved in a reaction,
   and the combining volume of X was V,
then consider only multiples of V as possibilities.
```

Given this assumption (and knowledge of the combining volumes: two parts by volume of hydrogen plus one part of oxygen yields two parts of water vapor), the program (let us call it DALTON') instead postulates two molecules of hydrogen and water (and if this was later found to be unsatisfactory, four and then six), while retaining the assumption of one oxygen molecule. Thus, at the third level in the search tree, DALTON' has the partially specified model ((H) (H) (O) → (W) (W)).

At this point the revised system moves to consider the internal structure of the hydrogen molecule, assuming it is composed of a single atom (say h), and makes a similar assumption for oxygen. However, for the resulting model, ((h) (h) (o) → (W) (W)), there exists no decomposition of water in terms of h and o that satisfies the conservation assumption, so the program backs up and considers some other alternative. DALTON' next hypothesizes the oxygen molecule as composed of two atoms, and since this does allow conservation to be satisfied, a final model is constructed in which oxygen is diatomic and hydrogen is monatomic: ((h) (h) (o o) → (h o) (h o)). (These two search paths are shown on the right side of Figure 3.) While this model differs from the modern-day one, it is consistent with Guy-Lussac's data and encounters difficulty only when other reactions are considered. For example, the assumption that hydrogen is monatomic does not work for the ammonia reaction.

Like most of the programs we have described, DALTON is stated as a production system. In default mode, the system uses a few simple rules to formulate simpler models first, and more complicated ones as necessary. However, if new condition-action rules are added to the system, they take precedence over the default rules and can direct search down paths that might otherwise not be considered. Thus, one can insert a rule that would match if the combining volume of some substance is known, and use this information to determine the number of molecules used for that substance in the model. The conservation assumption is implemented in a similar fashion, so that it generates a molecular structure of a reaction's output that uses all particles occurring in the input.

Once DALTON has generated a successful model for a reaction, it converts this knowledge into productions. For instance, having arrived at the diatomic explanation of water given above, the program would store one rule concerning the molecules of hydrogen involved, another for oxygen molecules, and a third for the water molecules. If the system is asked to explain the water reaction at a later date, it will be able to recall the number of molecules without search. DALTON also constructs productions describing the internal structure of various molecules, and while this knowledge is useful in re-explaining the water reaction, it is useful in other cases as well. For instance, when asked to model the ammonia reaction, the system would immediately propose that hydrogen was monatomic, based on the success of this assumption in its model of water. None of the models incorporating this assumption satisfy conservation, so the system would back up, hypothesize that hydrogen is instead diatomic, and eventually arrive at the correct model for ammonia.

However, DALTON must now also update its model of the water reaction. Since the system knows that its monatomic hydrogen rule was responsible for leading it astray, it removes this rule from memory and replaces it with a diatomic rule for hydrogen. It then focuses on the reaction that led it to construct the monatomic production, and checks to see if the replacement rule works here as well. In this case it does, producing the structure (h h o) for water, but had it run into difficulty, the process would have been repeated, with DALTON considering ever more complicated molecular structures (up to a limit), until both the water and the ammonia reaction had been successfully explained by a single rule.

## 5.4. Comments on DALTON

Although DALTON's methods are concerned with reactions, they are not limited to the chemical domain. For example, the field of elementary particle physics is also concerned with reactions, and with the formulation of structural models to explain those reactions. The most widely accepted theory in this domain accounts for the internal structure of protons, neutrons, and other hadrons in terms of a small set of hypothesized particles called *quarks*. In its present form, DALTON cannot rediscover the quark theory, but two relatively simple extensions should enable the system to arrive at the basic tenets of this framework.

First, the current version of DALTON requires knowledge of the components of a substance, or knowledge that a substance is elementary, such as hydrogen or oxygen. However, there are no directly observable "elements" in the field of particle physics, and in order to explain particle interactions, one must postulate entirely new substances that have never been seen. For example, the basic proton "molecule" is viewed as composed not of three proton "atoms", but as composed of two *u* quarks and one *d* quark. In order to regenerate the quark theory, DALTON must be modified to search the larger space of models in which such decompositions can occur. Alternately, one can imagine a modified version of STAHL capable of determining the unseen components of hadrons, with DALTON retaining its focus on the number of particles involved. In any case, the issue of inferred particles must be addressed in one system or the other.

An equally important aspect of the quantum theory involves the conservation of mass and of the various quantum numbers, such as spin and electric charge. In order to generate these features of the theory, DALTON must attempt to explain quantitative attributes of directly observable substances (such as protons and neutrons) in terms of attributes associated with inferred substances (such as quarks). Presumably, these constraints can be stated as theory-driven heuristics, much as the conservation of particles assumption is implemented in the current version. Once the system has been given this capability, it may also be able to rediscover the basic version of the caloric theory, in which the conserved properties of mass, heat, and heat quantity are used to explain changes in the non-conserved quantity temperature.

A less obvious application of DALTON involves the field of classical genetics. The hereditary rules for garden peas, first enumerated by Gregor Mendel in 1866, can be viewed as reactions in which characteristics of the parents are transformed into characteristics of the offspring. Given the first extension described above, along with a suitable replacement for the conservation assumption (since this does not apply in reproductive systems), DALTON should be able to arrive at the two-trait model originally formulated by Mendel. For example, let us suppose that the system is provided with genotypic statements of the result of inbreeding and crossbreeding, which might be induced by another discovery system (like GLAUBER) from phenotypic descriptions of these reactions. Then, if we let G stand for green peas that produce only green offspring, Y stand for yellow peas that produce only yellow offspring, and G' stand for green peas that produce mixed offspring, four basic reactions suffice to describe Mendel's observations: G G → G, Y Y → Y, G Y → G', and G' G' → G G' Y. Given these reactions, an extended version of DALTON should be able to infer that two primitive traits (say g and y) are required, and to decide that the genotype G can be modeled by the "molecular" pair (g g), that Y can be modeled by the pair (y y), and that G' can be modeled by the pair (g y). As we envision it, the system's explanation of these reactions would not involve the notion of dominance, nor would it predict the proportions in which the various genotypes are observed, but it would account for the basic qualitative relations between parents and offspring.

Before closing our discussion of DALTON, we should examine briefly its relation to DENDRAL [11], a well-known AI system that was also concerned with formulating structural models of substances, in this case complex organic molecules. Rather than using reactions for its basic information, DENDRAL searched for models that would account for mass spectrogram data. In addition, the system employed considerable knowledge of organic chemistry to direct its search through the space of possible models. There is no doubt that this early program could effectively search spaces in which DALTON would be quickly overwhelmed, and could generate structural models more complex than our system could begin to consider. However, this observation misses an important point. DENDRAL was concerned primarily with imitating 20th Century organic chemists who draw upon centuries of accumulated knowledge about chemicals and their reactions. In contrast, DALTON is concerned with an earlier stage in the discovery process, such as we find with the early chemists in their attempt to formulate atomic models with very little available knowledge. Thus, DALTON and DENDRAL can be viewed as lying at two ends of a spectrum, with the first studying simple discoveries in a knowledge-poor environment and the second focusing on more complex discoveries in a knowledge-rich environment. Ultimately, we may understand both approaches as special cases of a more general method for creating structural models, but that remains a topic for future research.

## 6. Towards an Integrated Discovery System

In the preceding pages, we examined four AI systems that address different aspects of the discovery process. While each of these programs is interesting in its own right, they should ultimately be combined into a single, integrated discovery system. One advantage of this approach is that it will increase our understanding of the relations among the various forms of discovery. In turn, this understanding will constrain the component systems, since the outputs of one program would have to conform to the input requirements of another. This will lead to revisions of the existing systems, and more robust and plausible discovery programs will result. Another benefit is that the resulting system would be more self-contained, relying less on the programmer and more on its own devices. To the extent this can be achieved, an integrated discovery system would be much less susceptible to the criticism that one is "building in discoveries" by providing the necessary inputs.

Since the notion of search is central to all four discovery systems, let us explore the role of search in the proposed integrated system. Clearly, the operators used by each of the systems will remain the same, as will the heuristics for applying those operators. The initial states for each system will be largely the same, but will no longer be provided by the programmer. Instead, they will be generated by other systems as output. Given a set of operators and rules for applying those operators, the specification of an initial state effectively defines a problem space. Thus, to the extent that discovery system A's initial state is created by another system B, we can claim that B has defined the problem space that A will search. This may lead A to specify a new initial state for B, thus defining a new space for it to search. The dream of the AI learning system that "pulls itself up by its own bootstraps" is an old one, and we do not expect it to be achieved in the near future. However, we do believe that it lies in the direction we propose to explore, in which individual learning systems are combined to form a whole that becomes greater than the sum of its parts.

In this section, we examine some scenarios in which significant interactions might take place among BACON, GLAUBER, STAHL, and DALTON. In each case, we will treat the individual systems as black boxes, and focus on the relation between their inputs and outputs. Although we are far from actually combining these programs into a unified system, we hope that these examples will convince the reader that such a system is not only possible but necessary if we ever hope to understand the complex process we call scientific discovery.

## 6.1. Designing Experiments and Generalizing Laws

Earlier in the paper, we noted that the discovery of qualitative laws often precedes the discovery of quantitative relations. This suggests that GLAUBER should be able to contribute something to BACON's search for numeric laws. However, the most obvious connection involves the search through the space of *data* rather than the space of laws. The reader will recall that BACON relies on the programmer to provide a set of variables and values, leading the system to run particular factorial experiments. Our hope is that GLAUBER will give BACON enough information to let it design its own experiments. For instance, suppose BACON were told by GLAUBER that nitric oxide, nitrous oxide, and nitrogen dioxide were all substances that resulted from reactions between nitrogen and oxygen. Given knowledge of this class of compounds, an extended version of BACON might design an experiment in which the substances entering a reaction (oxygen and nitrogen) were held constant, while the output of the reaction was varied. If quantitative variables such as the weights of the substances were examined, the resulting experiment would lead BACON to Dalton's law of multiple proportions, as described in an earlier section.

The second use of GLAUBER's output relates to BACON's generalization process. As it is currently implemented, BACON initially associates intrinsic values with all potentially relevant symbolic conditions, and generalizes by removing conditions whenever it finds that a set of intrinsic values is useful in a new context. However, the availability of the classes generated by GLAUBER presents an alternate generalization method. Rather than removing conditions entirely, one can generalize by replacing the symbol in a condition with the class containing that symbol. For instance, suppose BACON has stored a set of intrinsic values, with one condition for retrieval being that one of the substances entering the reaction is HCl. Next, suppose that the system finds the same intrinsic values useful when the substance is $HNO_3$ instead of HCl. Rather than inferring that this condition is irrelevant, BACON might decide that the intrinsic values should be retrieved whenever an *acid* is involved in the reaction (provided that GLAUBER had already defined this concept). This is a more conservative approach to generalization, and would allow BACON to express a larger class of hypotheses than it currently can. Of course, the system could eventually decide to remove this condition entirely, should the intrinsic values prove useful for non-acids as well.

This approach to generalization suggests that GLAUBER might find a use for BACON's output as well. Imagine an alternate scheme for generalizing intrinsic values, in which BACON iterates through all symbolic values, collecting those for which a set of intrinsic values are useful. Suppose the connection between symbols and values is stored in propositions, such as (intrinsics of {HCl} are {1.23 2.76 4.35}) and (intrinsics of {$HNO_3$} are {1.23 2.76 4.35}). Given such a set of propositions, GLAUBER could define a class (say A) based on those substances for which the values were useful, and formulate a law summarizing this knowledge, such as (intrinsics of {A} are {1.23 2.76 4.35}). If this class corresponded to another class, such as acids, so much the better. Thus, one can imagine GLAUBER aiding BACON's generalization process, or BACON's generalization method providing data for GLAUBER's discoveries, depending on which system is allowed to operate first.

## 6.2. Determining the Components of Acids

The fact that both STAHL and GLAUBER are capable of dealing with reactions between substances suggests that there is room for interaction between these systems. If GLAUBER is given reactions such as (reacts inputs {HCl NaOH} outputs {NaCl}) as inputs, it generates abstract reactions like (reacts inputs {acid alkali} outputs {salt}) as output. If such laws are passed to STAHL as data, the program will attempt to determine the components of the "substances" involved. In this case, the system would infer that all salts are composed of an acid and an alkali. This conclusion is not very surprising, though it is an inference one would like a discovery system to make.

More complex interactions become possible when one realizes that concepts such as HCl and NaOH are not primitive at all, but are based upon lower level observations much like the higher level concepts of acid and alkali. For instance, GLAUBER might be given many facts concerning the taste and color of a large set of substances (let us call them o1, o2, and so forth). Some of these substances would have very similar tastes, as well as very similar colors. Based on such shared properties, these chemicals would be grouped into the classes we know as hydrogen (H), chlorine (Cl), and others. If the primitive substances had been involved in reactions such as (reacts inputs {o1 o2} outputs {o3}), GLAUBER would rewrite these in terms of the new classes, giving reaction "laws" like (reacts inputs {H Cl} outputs {HCl}). Such laws would then be processed by GLAUBER to determine still higher level classes and laws. However, they could also be passed as inputs to the STAHL system.

Given inputs such as (reacts inputs {H Cl} outputs {HCl}), STAHL would apply its rules to infer the components of the substances involved. In this case, it would immediately infer that HCl is composed of hydrogen and chlorine. By itself, this inference is not very interesting. However, suppose STAHL then passed this result back to GLAUBER as additional data. In order to do this, it must represent the inference in GLAUBER's terms, but the existing (components of {HCl} are {H Cl}) will serve quite well. Given this fact and similar facts, such as (components of {$HNO_3$} are {H $NO_3$}), and given examples of reactions involving acids and alkalis, GLAUBER would formulate the class of acids, and generate by substitution the laws (components of {acid} are {H Cl}) and (components of {acid} are {H $NO_3$}). Taken together, these laws would lead to a new class (let us say acid-components) with members like Cl and $NO_3$, along with the law (components of {acid} are {H acid-component}). This law (appropriately quantified) states that all acids have hydrogen as one their components. This conclusion can be reached through a complex interaction in which GLAUBER affects STAHL's search through the space of componential models, and STAHL in turn affects GLAUBER search through the space of classes and qualitative laws. A similar line of reasoning would lead the GLAUBER/STAHL combination to the conclusion that all metals have phlogiston as one of their components.

## 6.3. Building Structural Models

As we have seen, STAHL focuses on determining the *components* of chemical substances, while DALTON is concerned with the *number* of particles involved in a reaction. Thus, STAHL can be viewed as laying the groundwork for a detailed structural model, with DALTON being responsible for finalizing the model. Moreover, DALTON requires knowledge about the components of a substance in testing its conservation assumption, and it is natural to assume that this information comes from STAHL. In fact, the coupling between these programs is already sufficiently close to view them as successive stages of a single system, and we expect to merge them in our future research. Let us explore the form such a combined system might take.

One can identify three distinct stages in the process of building structural models. The first involves identifying the components of substances, which is the focus of the STAHL system. We have discussed some potential extensions of this system, such as providing the ability to postulate unobserved components, but this would not alter the basic goal of the system. The second stage involves determining the number of times each component occurs in some substance, which is the focus of DALTON. Again, we have discussed some possible extensions, such as determining numeric attributes of the components, but the basic task remains the same. The final stage, which we have so far ignored, involves specifying the manner in which the various components are *connected* to each other. Early chemists were able to avoid this issue, but the discovery of organic molecules eventually forced them to deal with the problem. Kekule's insight about the structure of the benzene ring was essentially an insight about the connections between the components of that compound. Search in this stage would involve selecting a pair of components to connect, and selecting a type of bond to connect them.

We envision a single discovery system that searches the space of structural models, first determining the components involved, then identifying the number of particles taking part, and finally modeling the connections between these particles. Starting with completely abstract models, this system would successively instantiate them until their complete structure had been determined. At each stage in this instantiation process, the system would employ constraints, such as the conservation assumption, to reject some models in favor of others. Although the space of models would be quite large, search through this space would be relatively constrained. Although considerable work would be involved in constructing such a program, it would be an important step toward integrating the four discovery systems we have described.

## 6.4. Discovering the Principles of Inheritance

Earlier in the paper, we outlined an extended version of GLAUBER that would be able to note patterns among conjunctions of facts. We discussed the application of this system to Mendel's data on heredity, showing how it could be used to infer genotypic classes (e.g., pure-breeding green peas G, mixed green peas G', and pure-breeding yellow peas Y) from observations about phenotypes (e.g., green and yellow peas). In another section, we proposed an extended version of DALTON that, given genotypic descriptions of the offspring resulting from various matings, would be able to infer Mendel's two-trait model to account for those descriptions.[13] This suggests a straightforward relation between the two programs that should extend to other domains besides genetics. We see GLAUBER starting with directly observed reactions and, based on regularities among those reactions, rewriting them at a higher level of description. DALTON would then take the higher level reactions, and devise structural models to account for them. According to this view, GLAUBER would serve mainly as a preprocessor for DALTON, transforming direct observations into an initial state that the structural modeler could operate upon.

However, information can flow in the opposite direction as well. Once DALTON has constructed models for a set of genotypic classes (such as {g g} for G, {y y} for Y, and {g y} for G'), this information could be passed back to GLAUBER. For instance, suppose GLAUBER begins with the following knowledge, some of which would be provided by DALTON:

(components of {Y} are {y y})
(components of {G} are {g g})
(components of {G'} are {g y})
(has-property object {Y} color {yellow})
(has-property object {G} color {green})
(has-property object {G'} color {green})

Given this information, our extended version of GLAUBER would note two "facts" involving green-

[13] In fact, this could best be accomplished by the integrated version of STAHL and DALTON just described.

colored classes, and that both of these classes (G and G') has the symbol g as one of their components. As a result, the following two laws would be formulated:

$\forall Q\ \exists P$ (components of {Q} are {g P})
$\forall Q$ (has-property object {Q} color {green})

In addition, the class Q is defined as the union of the classes G and G', while P is defined as having the members y and g.[14] Taken together, these laws state that all green peas contain at least one instance of g in their list of components. This example is similar to the earlier case in which GLAUBER noted hydrogen as a component of acids, but one can interpret it somewhat differently. In the context of genetics, the above law is stating that g is a *dominant* trait, since it leads to green plants whenever it occurs as a component. Again, we have seen that complex feedback between two discovery methods can lead to laws that could not be discovered by either method alone.

## 6.5. Constraining the Search for Structural Models

We have seen how DALTON's search through the space of structural models can be altered by heuristics, such as the combining volume rule that led to Avogadro's model of the water reaction. However, we have not discussed the origin of the information used by such rules. For instance, Avogadro's heuristic must know the combining volumes for a reaction before it can be used to constrain search. Since this information is numeric, it is natural to consider BACON as a possible source, and upon reviewing BACON's chemical discoveries, we find that the system's common divisor method generates the combining volumes required by DALTON. Thus, BACON's output can be used to direct DALTON's search through the space of possible models.

We have discussed an extended version of DALTON which determines numeric properties of the components in its models. For example, the system might estimate the relative atomic weights of elements taking part in a set of reactions. (This was a major concern of the early chemists.) Given such estimates, one can imagine DALTON placing additional constraints on its models, and using these constraints to reject some models in favor of others. For example, the system might require that the estimated atomic weights be consistent across different reactions. However, in order to estimate the relative weights of the components in a model, DALTON would have to know the combining weights of the substances involved in a set of reactions. Again, BACON is the obvious source for such knowledge, since it generates combining weights at the same time it produces combining volumes. In summary, BACON has the potential to place significant constraints on DALTON's search process. It is interesting to observe that data-driven methods, like those used in BACON, can be such an aid to theory-driven behavior of the type found in DALTON.

## 6.6. Structure of the Proposed System

The above scenarios provide some idea of the *behavior* we expect from the integrated discovery system, but we have not discussed the *structure* of the proposed system. In particular, we should consider how closely linked the systems will be to one another. In considering the relation between STAHL and DALTON, we decided that the coupling should be very close, since these systems can actually be viewed as dealing with different stages in the same search process. However, it is not clear that the same conclusion holds for BACON, GLAUBER, and STAHL/DALTON, since these systems seem to address genuinely different aspects of discovery – the search for quantitative laws, the search for qualitative laws, and the search for structural models. More likely, the systems should be given access to a common blackboard, and care should be taken to ensure compatible representations.

[14]The current version of the system cannot handle situations in which a substance like g is treated as both a constant and a member of a class. This capability would have to be added before GLAUBER could work as proposed.

If we assume that the systems should be loosely coupled, we must still specify whether interaction occurs occasionally or continuously. The first approach assumes that one system would begin, run its course, and then deposit its results on the common blackboard, to be followed by another system which takes advantage of these results to define its search space. This fits in well with the current version of GLAUBER, which requires all facts at the outset of a run. An alternate scheme would have the systems running concurrently, with each depositing results on the blackboard, and with these results dynamically affecting the paths taken by other systems. This approach is well-suited to the STAHL program, which already uses an incremental approach to formulating componential models. Although an incremental system like STAHL (and to some extent BACON) can be provided with all the data at the outset, an all-at-once system like GLAUBER cannot be run in incremental mode. Thus, if we decide to pursue an incremental version of the integrated discovery system, GLAUBER will have to be substantially revised in order to fit into this framework.

## 6.7. Conclusions

In this paper, we examined four aspects of the diverse activity known as scientific discovery – finding quantitative laws, generating qualitative laws, inferring the components of substances, and formulating structural models. Our approach involved constructing AI systems that focused on these different facets of science, and testing them on their ability to replicate historical discoveries. We drew our examples mainly from the history of chemistry, since this area provided useful tests for each of the systems, and since it allowed us to explore potential connections between the discovery programs. We found that each of the systems could be usefully viewed as carrying out search through a space of laws or models, and we examined the operators and heuristics used to direct search through these spaces. We also found that each of the systems has some important limitations, and proposed some extensions that should lead to improved future versions.

Although the four systems – BACON, GLAUBER, STAHL, and DALTON – have each contributed to our understanding of discovery, we believe that an even greater understanding could result from exploring the *relations* among the systems. As a result, we plan to construct an integrated discovery system that will incorporate the individual systems as components. As we have noted many times, scientific discovery is a multi-faceted process, and even within such an expanded framework, we must omit many of its important aspects. For instance, we have not addressed the formulation of mechanistic explanations such as the kinetic theory of gases, the role of structural analogies as studied by Winston [12] and Gentner [13], or the design of new measurement devices. Thus, even our goal of an integrated discovery system is limited in some important respects. However, limiting one's focus of attention is a venerable and useful tradition in the history of science, and there will be ample time to incorporate these additional facets of discovery after we better understand the relations among the four existing systems.

## References

1. Newell, A. and Simon, H. A.. *Human Problem Solving.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1972.

2. Langley, P., Bradshaw, G., and Simon, H. A. Data-driven and expectation-driven discovery of empirical laws. Proceedings of the Fourth National Conference of the Canadian Society for Computational Studies of Intelligence, 1982, pp. 137-143.

3. Rosenbloom, P. *The Chunking Model of Goal Hierarchies: A Model of Practice and Stimulus-Response Compatibility.* Ph.D. Th., Department of Computer Science, Carnegie-Mellon University, 1983.

4. Bradshaw, G., Langley, P., and Simon, H. A. BACON.4: The discovery of intrinsic properties. Proceedings of the Third National Conference of the Canadian Society for Computational Studies of Intelligence, 1980, pp. 19-25.

5. Langley, P., Bradshaw, G., and Simon, H. A. Rediscovering chemistry with the BACON system. In *Machine Learning: An Artificial Intelligence Approach,* R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds., Tioga Press, Palo Alto, CA, 1983.

6. Langley, P., Zytkow, J., Bradshaw, G. and Simon, H. A. Mechanisms for qualitative and quantitative discovery. Proceedings of the International Machine Learning Workshop, 1983, pp. 121-132.

7. Wolff, J. G. Grammar discovery as data compression. Proceedings of the AISB/GI Conference on Artificial Intelligence, 1978, pp. 375-379.

8. Michalski, R. S. and Stepp, R. E. Learning from observation: Conceptual clustering. In *Machine Learning: An Artificial Intelligence Approach,* R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds., Tioga Press, Palo Alto, CA, 1983.

9. Brown, J. S. Steps toward automatic theory formation. Proceedings of the Third International Joint Conference on Artificial Intelligence, 1973, pp. 20-23.

10. Emde, W., Habel, C. H., and Rollinger, C. The discovery of the equator or concept driven learning. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 1983, pp. 455-458.

11. Feigenbaum, E. A., Buchanan, B. G., and Lederberg, J. On generality and problem solving: A case study using the DENDRAL program. In *Machine Intelligence 6,* Edinburgh University Press, Edinburgh, 1971.

12. Winston, P. H. "Learning and reasoning by analogy." *Communications of the ACM 23* (1980), 689-703.

13. Gentner, D. "Structure mapping: A theoretical framework for analogy." *Cognitive Science 7* (1983), 155-170.