Notice: This Material
may be protected
by Copyright Law
(Title 17 U.S.C.)

Z
699
C3
no.89-20
c.2

AN INTEGRATED APPROACH TO EMPIRICAL DISCOVERY

Bernd Nordhausen (bernd@ics.uci.edu)
Pat Langley (langley@ics.uci.edu)
Department of Information & Computer Science
University of California, Irvine, CA 92717

Technical Report 89–20

26 July 1989

The ideas in this paper have resulted from discussions with members of the UCI machine learning group. We would particularly like to thank Randy Jones and Don Rose for their contributions, as well as Jeff Shrager, who gave valuable comments on drafts of this paper.

To appear in J. Shrager & P. Langley (Eds.), *Computational Models of Discovery and Theory Formation.*

This work was supported in part by Contract N00014–84–K–0345 from the Computer Science Division, Office of Naval Research. The first author was also supported by a Regent's Dissertation Fellowship from the University of California.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report No. 13	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* An Integrated Approach to Empirical Discovery		5. TYPE OF REPORT & PERIOD COVERED Interim Report 7/88–3/89
		6. PERFORMING ORG. REPORT NUMBER UCI–ICS Technical Report 89–20
7. AUTHOR*(s)* Bernd Nordhausen Pat Langley		8. CONTRACT OR GRANT NUMBER*(s)* N00014–84–K–0345
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Information & Computer Science University of California, Irvine, CA 92717		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Computer Science Division Office of Naval Research Arlington, Virginia 22217		12. REPORT DATE 1 August 1989 13. NUMBER OF PAGES 25
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 J. Shrager & P. Langley (Eds.), *Computational Models of Discovery and Theory Formation.*

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

machine learning
conceptual clustering
taxonomy formation
qualitative discovery
quantitative discovery
empirical laws

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

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. ABSTRACT

In recent years, researchers in machine learning have investigated three main aspects of empirical discovery: forming taxonomies, finding qualitative laws, and finding quantitative laws. In this paper, we introduce IDS (Integrated Discovery System), a system that integrates these aspects of scientific discovery. We begin by examining the system's representation, showing how it describes events like chemical reactions as sequences of qualitative states. IDS incrementally processes these sequences to build a hierarchy of states, forming qualitative laws and numeric relations to augment this hierarchy. We discuss the three learning components that produce this behavior, using examples to illustrate the processes. Since this work is still in progress, we describe the current status of IDS and close with our plans for extending the system.

1. Introduction

Philosophers of science distinguish between two forms of discovery – the generation of *empirical laws* and the formation of *theories* (Thagard, 1988). The first activity involves descriptive generalizations that summarize observations such as Ohm's law and the ideal gas law. The second concerns the explanation of phenomena, which often involves postulating unobserved structures or processes. Examples of scientific theories include the fluid model of electricity and the kinetic theory of gases. Of course, science also involves many other components, including the design of experiments and measuring instruments, but it is often useful to focus one's attention on a limited set of phenomena. In this paper, we focus on the discovery of empirical laws.

In recent years, researchers in machine learning have investigated three main aspects of empirical discovery. The first relates to the process of taxonomy formation. Before one can formulate laws, one must first establish the basic concepts or categories that one hopes to relate. For instance, one might group certain substances together as acids, alkalis, or salts depending on their tastes. Research on *conceptual clustering* (Michalski & Stepp, 1983; Lebowitz, 1987; Fisher, 1987) addresses this problem, even though it has seldom been cast as relevant to scientific discovery. This task involves organizing a set of observations into a conceptual hierarchy, which can then be used to classify new observations. Fisher and Langley (1985) review work on conceptual clustering and its relations to statistics, whereas Gennari, Langley, and Fisher (in press) examine incremental clustering methods.

Another facet of empirical discovery concerns the generation of qualitative laws. In this case, the goal is to uncover qualitative relations that hold across a set of observations. Thus, one might note that acids tend to react with alkalis, and that the result is always some salt. Researchers who have addressed this problem include Brown (1973), Lenat (1977), Emde, Habel, and Rollinger (1983), Langley, Zytkow, Simon, and Bradshaw (1986), Jones (1986), and Wrobel (1988).

Finally, empirical discovery can involve the production of quantitative laws. Here the goal is to find mathematical relations between numeric variables. For instance, one might determine the amount of hydrochloric acid that combines with a unit amount of sodium hydroxide, and also note the amount of sodium chloride that results from this reaction. Researchers have developed a number of AI systems that have rediscovered a variety of numeric laws from physics and chemistry (Langley, Simon, Bradshaw, & Zytkow, 1987; Falkenhainer & Michalski, 1986; Kokar, 1986; Zytkow, 1987). In some cases, these systems find not only quantitative relations but also qualitative conditions on the laws.

Although each of these systems is successful at its specific task, no system to date has attempted to integrate these different aspects of science. The goal of our research is to develop a framework that unifies all three components of empirical discovery. To this end, we have developed IDS, an integrated discovery system that incorporates mechanisms for taxonomy formation, qualitative discovery, and numeric discovery. In the following pages we describe the IDS system in detail. The next section describes the representation and organization of knowledge, and Section 3 presents the main discovery components and their relation to each other. Both sections contain examples to clarify the system's structures and

processes. We close the chapter by describing the status of IDS, along with our plans for evaluating and extending the system.

2. Representation and Organization in IDS

IDS' representation draws upon recent work in qualitative physics, describing its observations as sequences of qualitative states. The system also uses this notation in stating its taxonomy and laws, which require some organization of memory. On this dimension, IDS borrows from recent work on incremental approaches to conceptual clustering. In this section, we give the details of representation and organization, first dealing with IDS' inputs and then with its outputs.

2.1 Inputs to IDS

Every discovery system starts with some background knowledge, whether this bias is made explicit or not. In IDS, this takes the form of a simple domain theory that describes classes of objects the system may encounter. This knowledge is represented as an *is-a* hierarchy, similar to what Michalski (1983) calls a 'structured descriptor' and what Mitchell, Utgoff and Banerji (1983) call a 'concept description grammar'. Figure 1 presents an example of such a hierarchy for certain of chemical substances. Although this example involves a disjoint hierarchy, the system can also handle nondisjoint structures in which nodes can have multiple parents. For instance, one might know that the substance HCl comes in two different colors, either green or blue.

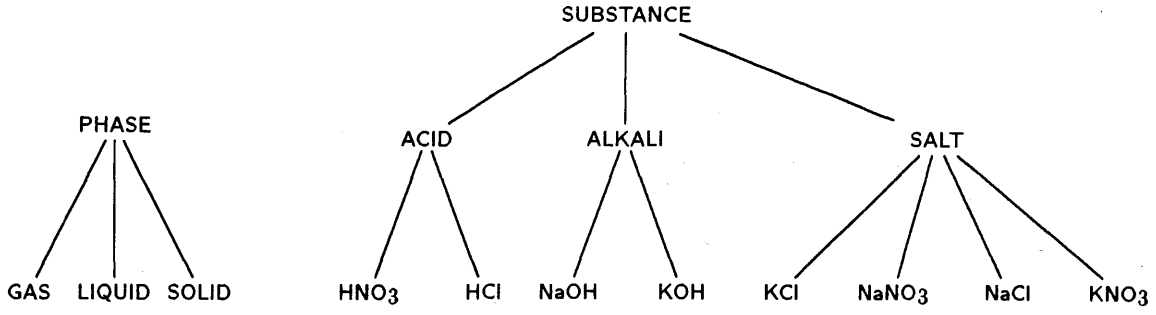

Figure 1. A simple domain theory for chemical substances.

An empirical discovery system also requires some data or observations on which to base its generalizations. In IDS, this information is represented as *histories* (Hayes, 1979), which are sequences of qualitative states that the system observes in an incremental fashion. Each 'state' represents an interval of time during which objects exhibit 'constant' behavior; this representation borrows heavily from Forbus' (1985) qualitative process theory.

Figure 2 shows the history for a simple chemical reaction with three distinct qualitative states. The initial state contains two separate objects, liquid HCl and liquid NaOH. When these substances are combined by an external agent, a new state begins that contains three

objects – the two original reactants and a new product.[1] During this state, the masses of the reactants are decreasing, whereas the mass of product is increasing. This is a form of constant behavior, since the signs of the derivatives remain unchanged.

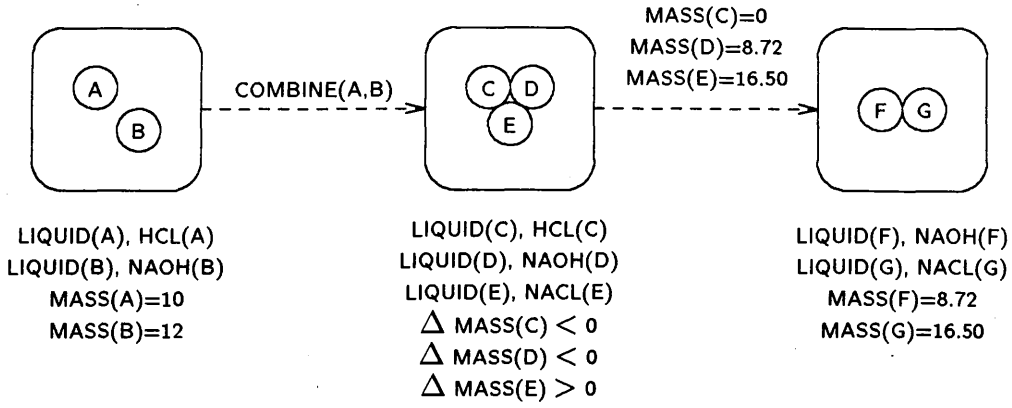

Figure 2. The sequence of qualitative states observed during a chemical reaction.

A qualitative state ends and a new one begins whenever the sign of any derivative changes; i.e., when any increase or decrease of a variable starts or stops.[2] A boundary between states also results when any structural change takes place. Both occur in Figure 2 when the mass of the HCl in State 2 reaches zero, for at this point one of the reactants disappears and the mass changes halt in the remaining objects. As a result, the history enters a new state in which these two objects are in contact but in which their masses remain constant. Although IDS is given these boundaries, one can imagine a system that found them on its own (Kuipers, 1985; Weld, 1986).

IDS represents each qualitative state as a frame with four slots. The *object description* slot describes the objects present in the state using the domain theory for objects. For example, object C in Figure 2 is described as liquid(C), HCl(C). A state also includes a *structural description*, such as touches(C, D). The *changes* slot contains a list of zero or more changes occurring in the state. As in Forbus' theory, we express changes in terms of derivatives. For instance, a decrease in the mass of C is expressed as Δ mass(C) < 0. Finally, the *quantity* slot describes numerical attributes that remain constant during a state.

Histories are inherently sequential, and IDS represents the successor of a given state using a *successor* link. These links can be labeled by a *transition condition*, which identifies the condition under which the current state ends and its successor begins. These transition conditions may involve either quantitative descriptions, such as mass(B) = 12, or the actions of an external agent, such as combine(A, B). For instance, the melting and boiling points of a substance are commonplace transition conditions.

IDS processes qualitative states one at a time, in the order they occur in a historical sequence. The system also receives the information on the temporal order of these states.

[1] We have omitted the second product, liquid H_2O, for the sake of simplicity. Also, note that the objects are labeled by pattern-match variables that are different from state to state.

[2] These state boundaries correspond to limit points in Forbus' qualitative process theory.

For example, it would first be given the initial state in Figure 2. After processing is complete, IDS is then presented with the second state along with the fact that this state is the successor of State 1. The third state is then given in the same fashion, with the information that it is the final state in the history. The system is next presented with the first state of the next history, and so on. As we will see, IDS uses these states and the temporal relations between them to incrementally form a hierarchy of qualitative states and to discover empirical laws.

2.2 Outputs of IDS

The incremental nature of IDS means the system has no explicit outputs, since it continues processing states as long as they are available. However, the system produces a knowledge structure after each experience, and one can view these structures as its 'output'. We will focus on three aspects of this output – the taxonomy, qualitative laws, and numeric laws.

IDS organizes the qualitative states it observes into a taxonomic hierarchy, with specific states as terminal nodes and with abstract states as internal nodes. This hierarchy takes the form of a tree, so that no specific state can belong to more than one abstract category. IDS does not make a distinction between instances and abstract states, thus the abstractions have the same slots as particular states (i.e., a description of the structure, the objects involved, the changes occurring during the state, and the constant quantities). The structure of the IDS hierarchy is similar in form to those generated by UNIMEM (Lebowitz, 1987) and COBWEB (Fisher, 1987), though these systems do not cluster qualitative states.

Figure 3 presents a top-level taxonomy that summarizes qualitative states involving various acids, alkalis, and salts, with solid lines standing for *is-a* links. For example, Node 4 describes cases in which a liquid alkali and a liquid salt are in contact with each other, whereas Node 5 specifies cases in which acidic and salty liquids occur. Node 3 is an abstraction of these cases in which the second substance is not specified. No changes are occuring in any of these three states, though changes are present in Node 2.

The IDS taxonomy connects states at varying levels of abstraction through *is-a* links, but histories also contain temporal information. Thus, the system also specifies *successor* links between nodes, indicating that one class of qualitative states follows another in time. Figure 3 also shows examples of *successor* links, using dashed arrows to indicate these temporal connections. For example, the link between Nodes 1 and 2 specifies that instances of Node 2 occur directly after instances of Node 1. Note that a given node may have several successors, some at different levels of abstraction.

Successor links may also specify the conditions under which the transition occurs. For instance, the label on the link between Nodes 1 and 2 – combine(A, B) – indicates that this transition occurs when the two objects in Node 1 are physically combined. Taken together, nodes and successor links represent qualitative laws similar in content to those found by GLAUBER (Langley et al., 1987). Thus, Node 1, Node 2, and the link between them asserts that when a liquid acid is combined with a liquid alkali, the two substances react to form

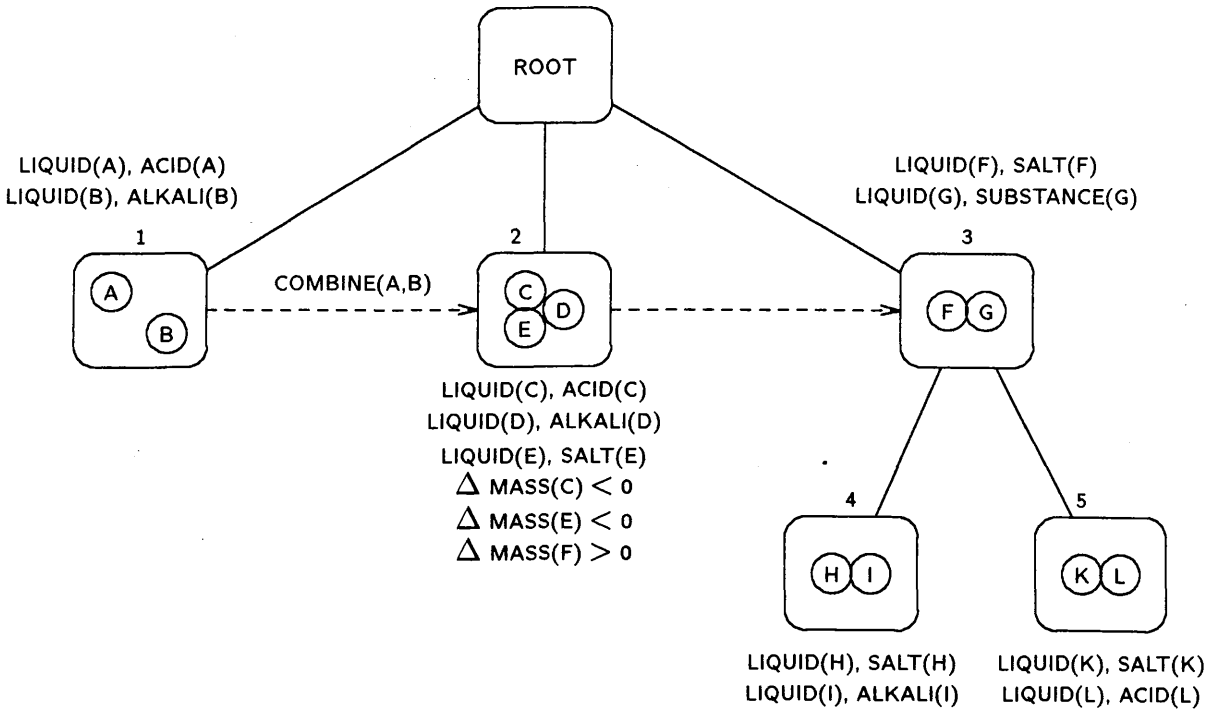

Figure 3. A taxonomy for acids, alkalis, and salts, augmented with *successor* links and transition conditions.

a salty liquid. This can be viewed as a restatement of the qualitative law 'acids react with alkalis to form salts'.[3]

Figure 4 shows the third aspect of IDS' output – the augmentation of nodes and successor links with numeric laws. One type of law describes the conditions for transitions between states. For instance, the successor link between Node 2 and Node 3 in this figure indicates that when the mass of the liquid HCl reaches zero, the reaction state (Node 2) ends and the final state (Node 3) begins. Nonzero values, such as the boiling point for a substance, may be stored as well. IDS also stores numeric laws with individual states that relate attributes within that state; the ideal gas law is one example of such a relationship.

In addition, the system forms numeric laws that relate attributes in different states in the same sequence. It stores these laws on *quantity relation* links that connect the states containing the related attributes. Figure 4 shows an example of such a cross-state law. This relation specifies that liquid HCl reacts with liquid NaOH in constant proportion to form liquid NaCl. The mass of the resulting NaCl is 1.64 times the initial mass of the HCl; this corresponds to the chemical concept of the *definite proportions* of the reaction.

The numeric laws generated by IDS are similar to those found by BACON (Langley, Bradshaw, & Simon, 1983), ABACUS (Falkenhainer & Michalski, 1986), and FAHRENHEIT

[3] We will not argue that this approach can represent *all* forms of qualitative laws, but we do feel that temporal relations among abstract qualitative states constitute an important subset of such laws.

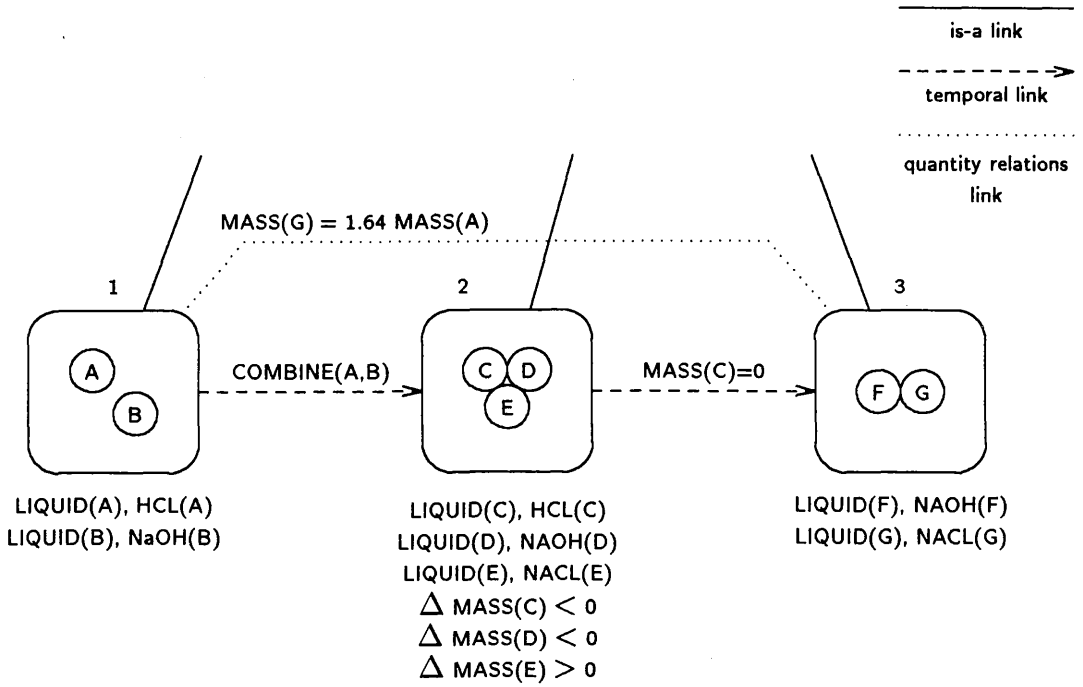

Figure 4. A numeric law relating attributes across states.

(Zytkow, 1987), but there is an important difference. These earlier systems found numeric laws and conditions for them, but their statement of the laws contained no information about the structural or physical context in which they occurred. Even as simple a relation as the ideal gas law actually involves a set of structurally-related objects that change over time. This is precisely the function of the taxonomy of qualitative states and the successor links in IDS. Unlike earlier approaches to numeric discovery, IDS describes a qualitative *context* for its quantitative laws.

3. Discovery and Prediction in IDS

Now that we have discussed IDS in terms of its inputs and outputs, we can present the mechanisms it uses for empirical discovery. Like much of the recent work in machine learning, the system can be characterized as using an incremental hill-climbing approach (Langley, Gennari, & Iba, 1987). Such systems process one instance at a time and do not reprocess substantial numbers of previous instances. Furthermore, these systems hold only one structure in memory and retain no information about their learning steps, so explicit backtracking cannot occur. Following this general strategy, IDS processes one qualitative state at a time, sorting the state through its current taxonomy, incorporating it into the hierarchy, and creating new nodes as needed. When a new node is created, IDS determines the successor of that node. If a new child does not obey a numeric law of its parent node,

Table 1
The IDS clustering algorithm

```
Variables: N, P, and Q are nodes in the hierarchy.
           I is an instance (a very specific node).
           X is a distance score between two nodes.

Cluster(N, I)
    For each child C of N
        compute the distance score between C and I.
    Let P be the node with the highest score.
    Let X be the score for placing I as a descendant of P.
    If X is sufficiently high,
       Then if P does not cover I,
               Then generalize P to cover I.
            Cluster(P, I).
       Else add I as a child of N.
            Merge-children(N, I).
Note:
  The distance measure is lexicographic, using states' slots
      to calculate the similarity between two states.
  The domain theory is used to generalize and merge states.
```

the system formulates an improved law. This cycle continues as long as the system receives new observations.

3.1 Forming a Taxonomic Hierarchy

Table 1 summarizes the IDS clustering algorithm, which has been heavily influenced by Lebowitz's (1987) work on UNIMEM and Fisher's (1987) work on COBWEB. When IDS receives a new qualitative state, it sorts this state through its hierarchy. Starting at the root node, the system computes the difference between the instance and each child of the current node. IDS measures these differences using a *lexicographical evaluation function* (Michalski, 1986). The total value of this function is computed from scores of similarity between the slots of the two states.[4]

The system then sorts the instance to the child that matched it most closely. If the match is sufficiently high (i.e., if the score is above a user-specified threshold), the selected child becomes the current node, and the sorting continues recursively. If the match with the selected child is high enough but that child does not cover the instance, the child is generalized so it covers the instance completely. If the match is not high enough, the instance is added as a new child of the current node.

Let us consider another example from the domain of alkalis and acids. Figure 5 shows a portion of an IDS taxonomy that describes the final states in a reaction sequence, after

[4] The structural description is treated as most important, followed by the description of changes, then by the object descriptions, and finally the descriptions of numeric attributes.

one of the initial substances has been completely used. The system receives a new instance (labeled 4 in the figure), which it sorts to Node 1. At this point, IDS computes the distance between the instance and the children of Node 1. In this case, Node 3 matches the instance more closely than Node 2. However, the score for placing the instance as a descendent of Node 3 is not above threshold, so the algorithm adds the instance as a new child of Node 1, as seen in Figure 6.

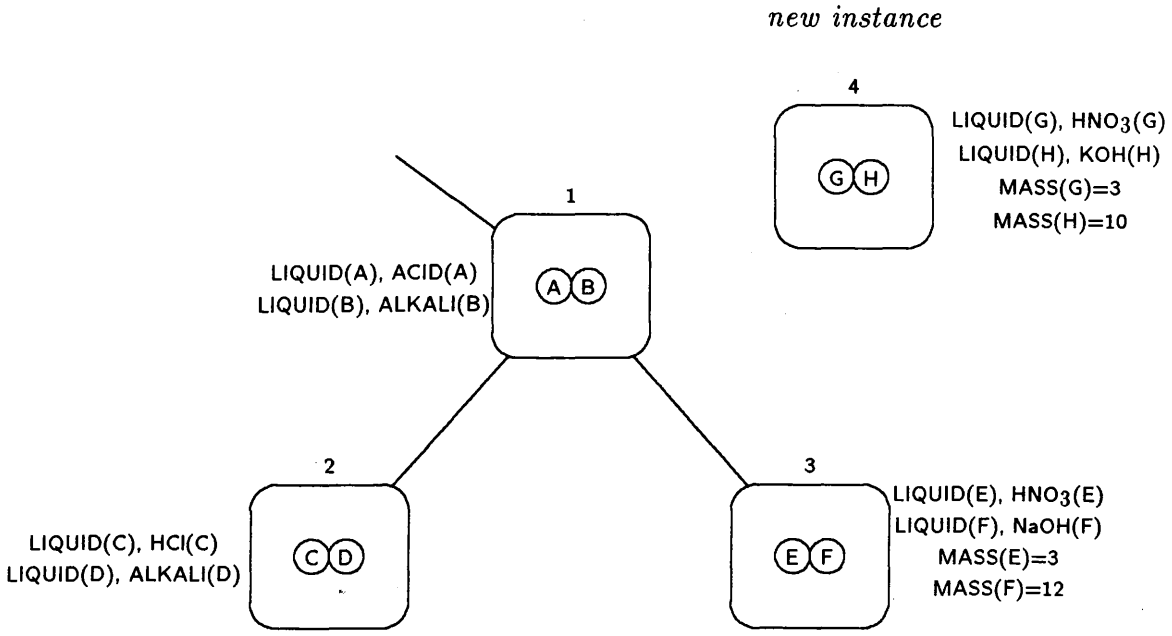

Figure 5. A taxonomy before incorporation of a new qualitative state.

As we noted earlier, IDS identifies the objects in each state using state-specific variables. In order to determine the best match between two states, the matcher generates all possible bindings between the variables in the two states. Using the evaluation function, the matcher then calculates a similarity score between the two states for each set of bindings and selects the one with the highest score. For example, there are two possible sets of bindings for the variables between Nodes 4 and 3 in Figure 7: {(G, E) (H, F)} and {(G, F) (H, E)}. Because the first binding set receives a higher score, the matcher concludes that G and H of Node 4 correspond to E and F of Node 3, respectively.

Whenever IDS adds a new instance as the child for a node in the hierarchy, it considers two ways to merge the node's children. First the system finds the two siblings that match the new child most closely. It then considers merging the new child with its closest sibling, as well as merging these two siblings.[5] IDS computes a score for each option using its evaluation function and merges the pair that produces the higher score, creating a merged (generalized) node that subsumes the pair. The system then stores the siblings as children

[5] For an optimal solution, the system would have to consider merging all possible pairs. However, informal experiments have shown that considering these two cases generally produces the desired results.

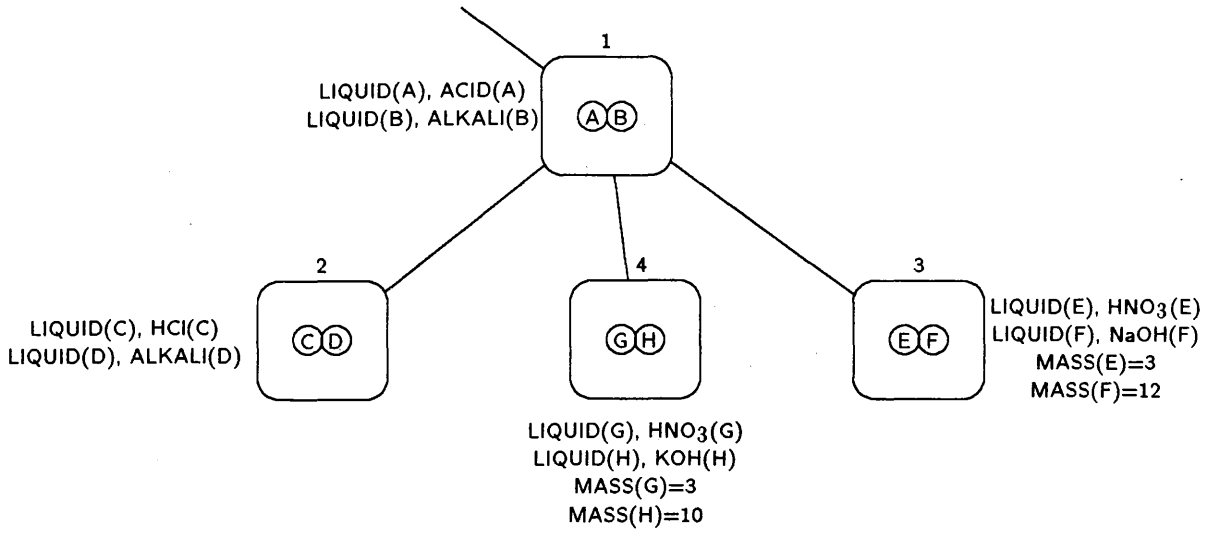

Figure 6. A taxonomy before two children have been merged.

of the merged node, which in turn is stored as a child of the original parent node. Merges of two nodes that result in a node identical to the original parent are not executed.

In our example, Node 4 (the new child) has only two siblings. Thus, the system considers two actions: merging Nodes 2 and 3, and merging Nodes 4 and 3. The second option receives the higher score, and since merging the two nodes does not produce a node that is identical to Node 1, the merge is carried out. The merged node (Node 5) is added as a child of Node 1, and Nodes 4 and 3 are added as children of Node 5. Figure 7 shows the modified taxonomy after merging has occurred, and Table 2 presents the algorithm for merging children.

The process for creating a generalized node is straightforward. IDS uses the matcher to find a correspondence between the variables in the merging nodes and the variables in the merged node, which it then uses to fill the slots of the new state. In general, each slot value in a new node is the intersection of that slot's values in the merging nodes. For example, given the situation in Figure 7, the matcher determines that variable I of Node 5 corresponds to G in Node 4 and E in Node 3, and that variable K in Node 5 corresponds to H and F. The quantity slot of Node 3 has a value of mass(E) = 3, mass(F) = 12, and the quantity slot of Node 4 has a value of mass(G) = 3, mass(H) = 10. Using the variable correspondence as a constraint, the intersection of these two sets is mass(I)=3, which becomes the value for the quantity slot in Node 5.

IDS computes the object descriptions of a merged state in a different manner, using the domain theory to determine the values of this slot. As described above, for each variable in the merged node, the matcher determines the corresponding variables in the merging nodes. The system then collects the components of the object description for each pair of corresponding variables. Next, IDS determines all closest common ancestors of each pair of components in the domain theory. These common ancestors become the components of the new object description for the variable in the merged node.

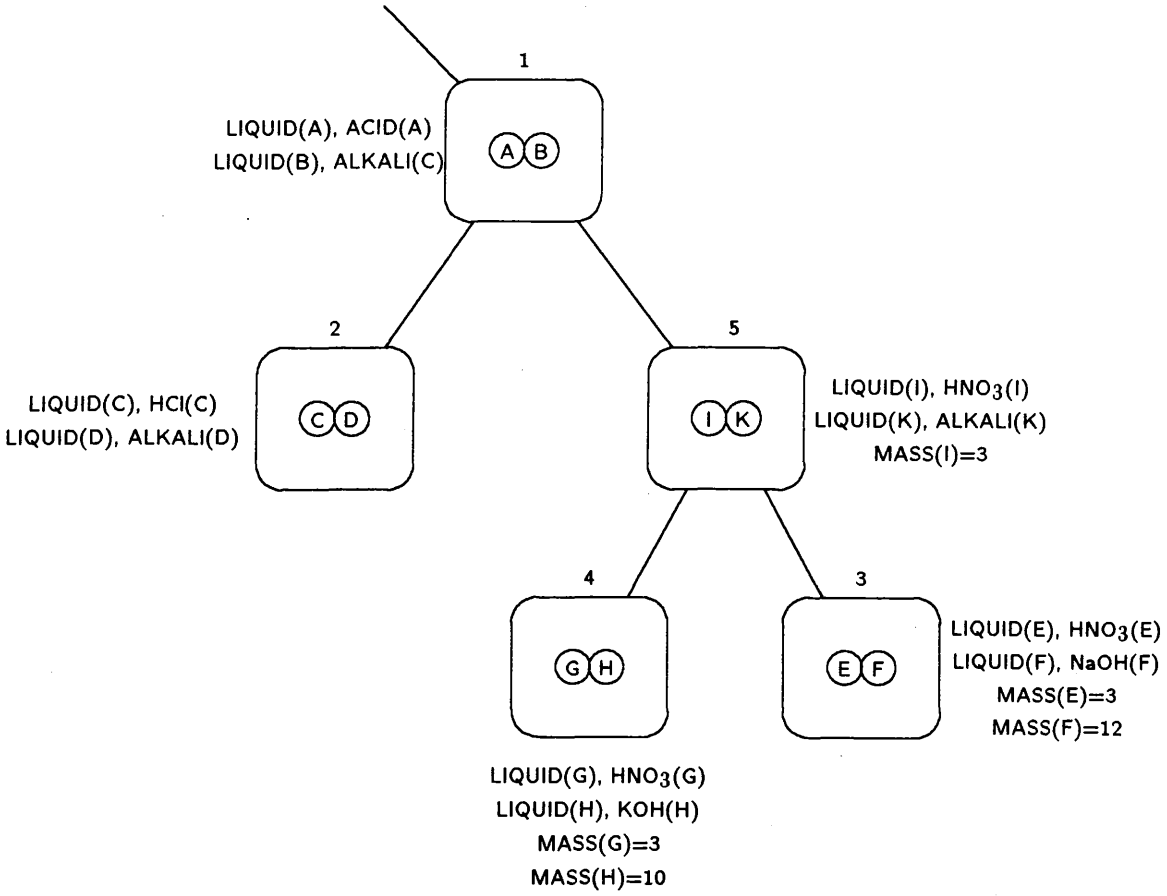

Figure 7. A taxonomy after two children have been merged.

We can clarify this procedure with an example. Given the situation in Figure 7, the matcher determines that variable H of Node 3 and F of Node 4 correspond to K in Node 5. The description components of H are liquid and KOH, whereas the components F are liquid and NaOH. The closest common ancestor of liquid and liquid (given the domain theory in Figure 1) is liquid, and the closest common ancestor of KOH and NaOH is alkali. The terms liquid and KOH have no common ancestor, nor do NaOH and liquid. Hence, the resulting description components for K are set to liquid and alkali.

As we noted above, IDS' clustering component has been influenced by Lebowitz' (1987) UNIMEM and Fisher's (1987) COBWEB, but there are some important differences. For instance, IDS and COBWEB form only disjoint taxonomies, in which each node has a single parent. In contrast, UNIMEM can sort an instance down multiple paths, producing a nondisjoint hierarchy. IDS differs from both earlier systems in that it includes no counts or probabilities on its features; each description in the taxonomy is categorical. Our system is probably most akin to COBWEB, though it uses a different evaluation function and lacks the latter's splitting operator (the inverse of the merge operator).

Lenat's (1982) AM system also organizes its concepts into a hierarchy and dynamically extends that hierarchy over time. However AM begins its existence with 250 heuristics

Table 2
The algorithm for merging children

```
Variables: N, P, Q, and R are nodes in the hierarchy.
           A is the newest child.
           X and Y are partition scores.

Merge-children(N, I)
    For each child C of N except A
        Compute the score of closeness between C and A.
    Let P be the node with the highest score.
    Let R be the node with the second highest score.
    Let X be the score of merging A and P.
    Let Y be the score of merging P and R.
    If X is the best score,
       Then let Q be the resulting node of merging P and A.
            If Q is not equal to N,
               Then place Q as a child of N.
                    Remove P and A as children of N.
                    Place P and A as children of Q.
    Else if Y is the best score,
         Then let Q be the resulting node of merging P and R.
              If Q is not equal to N,
                 Then place Q as a child of N.
                      Remove P and R as children of N.
                      Place P and R as children of Q.
```

and over 100 initial concepts, whereas IDS begins with a simple algorithm, a small domain theory, and an empty hierarchy. More important, Lenat's system generates new concepts by 'mutating' the definitions of existing ones and then testing them; we might call this an *exploratory* approach to discovery. In contrast, IDS (like UNIMEM and COBWEB) generates new concepts in direct response to observations, using a *data-driven* approach to discovery.

3.2 Discovering Qualitative Laws

We have seen that IDS represents qualitative laws in terms of abstract qualitative states and the successor links connecting them. With the exception of the final state in a history, every node in the taxonomy must have some successor node. This temporal information is given as part of the input, and this input specifies the links for terminal nodes in the hierarchy, but the system must induce the links between abstract nodes.

Whenever IDS forms a new abstract qualitative state (i.e., a nonterminal node in the hierarchy), it determines the successor for this new node. This is a simple process that involves finding the closest common ancestor of the successors of the new node's children.[6]

[6] This requires that all the successors of that node's children have been incorporated into the state hierarchy. Therefore, finding the closest common ancestor of a node is delayed until the

For an example, consider the partial taxonomy shown in Figure 8. Node 1 has two children, which are labeled Nodes 3 and 4. These nodes have the successors Nodes 5 and 6, respectively. In this case, IDS determines that the closest common ancestor of Nodes 5 and 6 is Node 2, and it asserts this node as the successor of Node 1.

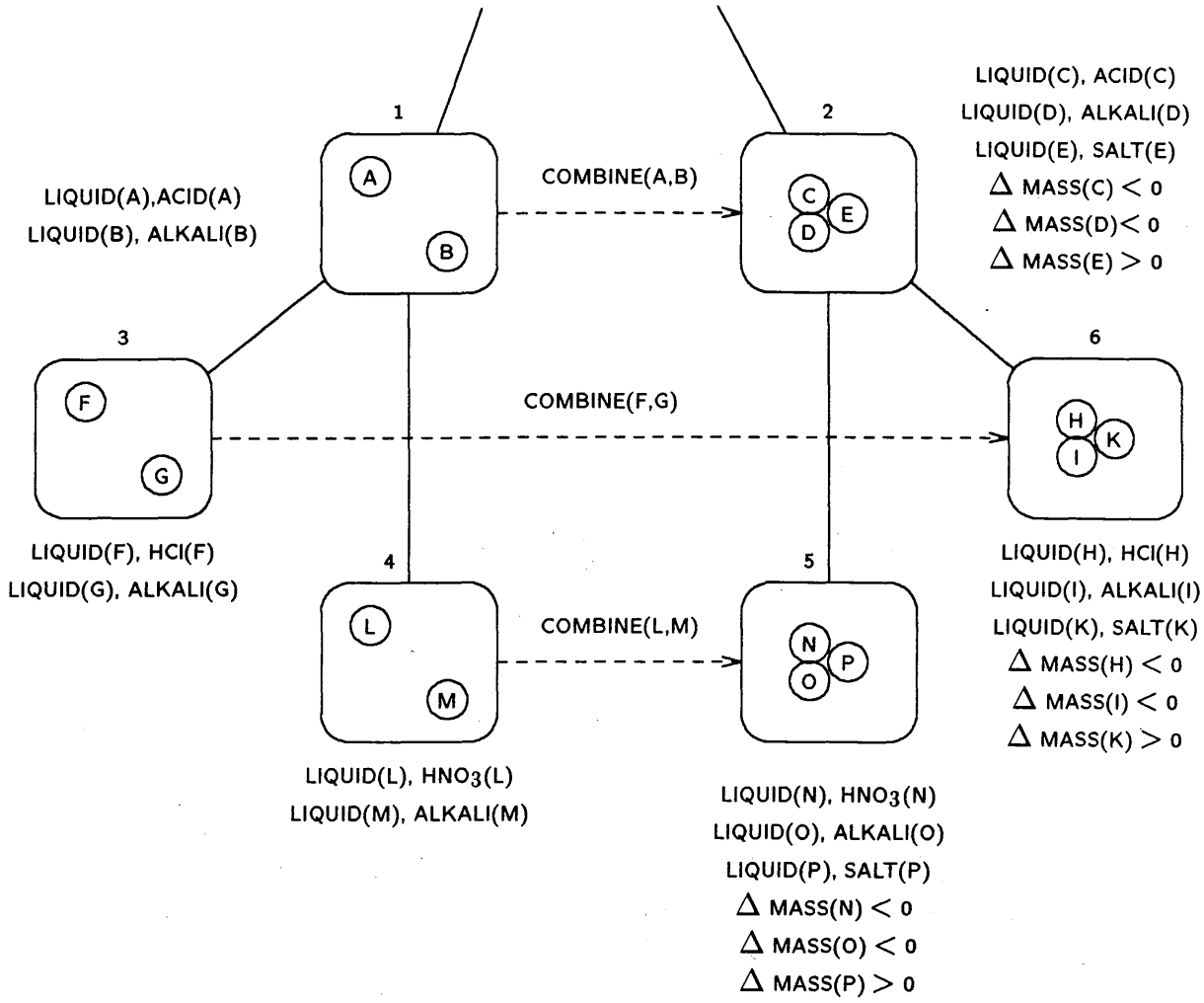

Figure 8. A qualitative law that results from the merging of two successor links.

In addition, IDS attempts to attach transition conditions to the new successor link, which may take the form of some external action or some quantity relation. The system determines these conditions in the same way that it forms merged nodes: by finding the structure common to the two links. For example, the transition formed between Node 1 and Node 2 is labeled **combine(A, B)**, because this action is stored on the successor link connecting Nodes 3 and 6, as well as that connecting Nodes 4 and 5. The act of adding

successors of all of its children have been clustered. Every set of nodes has at least one common ancestor – the root node. However, a successor link from a node to the root node in the hierarchy does not form a qualitative law with any useful content.

this condition is equivalent to inducing a law that states 'if a liquid acid is combined with a liquid alkali, they react to form a liquid salt'.

If IDS finds only some common actions in the children's successor links, it includes only the shared structure in the abstract link. If it can find no common structure, the system creates the successor link but specifies no transition conditions. In other cases, the conditions on the specific links involve numeric relations, such as reaching zero mass or achieving boiling point. In this situation, IDS attempts to find a numeric relation that covers the specific cases, using the algorithm described in the next section.

As we noted earlier, IDS formulates qualitative laws with similar content to those found by Langley et al.'s (1986) GLAUBER. However, the two systems arrive at these laws in very different manners. As we saw in the acid/alkali example, qualitative discovery in IDS is a simple process of finding two nodes' closest common ancestor. In contrast, GLAUBER spent considerable effort in finding classes of objects with common attribute values and playing similar roles.

3.3 Finding Numeric Laws

The third major component of IDS focuses on discovering numeric laws. As we saw earlier, these relations augment the qualitative descriptions, and they may specify the conditions for moving from one state to another, a relation between numeric attributes within a given state, or a quantitative relation between states. Each of these cases involves storing a law at a node or link in the taxonomy that summarizes information in the children of that node or link. IDS uses a single procedure to find all three forms of numeric law. Briefly, whenever the system adds a new child to an existing node in the hierarchy, it checks to see if the child obeys the laws currently stored at the parent. If it does not, IDS searches for new laws that cover the added child and its siblings.

For a given data set, the system conducts a beam search through the space of numeric terms to find a law that covers these data. More precisely, the search task can be stated as:

- *Given:* a set of base terms $a, b, c, \ldots$, along with one designated term (a) from that set;
- *Find:* a term $x = a^{n_0} \cdot b^{n_1} \cdot c^{n_2} \ldots$ such that a linear relation of the form $a = mx + n$ holds.

IDS searches from simple terms to more complex ones, using correlation analysis (Freund & Walpole, 1980) to direct the search process. As in BACON, the basic operators involve defining new terms as products and ratios of existing terms. The system initially examines correlations between the designated term and observable attributes, uses these to select promising products and ratios, and then recurses if it cannot find a law with the existing terms.

This search technique has a semi-incremental flavor. In cases where IDS has rejected an existing law, there is no need to reconsider the term in that law and those leading to the law. Thus, it uses the old term as the starting point for the new search, saving considerable effort over an approach that starts from scratch. However, this method does require one to store and reprocess all the data that led to the rejected law. As a result, it does not quite

Table 3
The algorithm for finding numeric laws

```
Variables: S is the set of base terms.
           D is the designated term.
           C is the set of current terms.
           P and Q are sets of terms.
           A is a defined term.

Find-law(D, S, C)

    Let A be the term in C that has the highest
         correlation with D.
    If the correlation between D and A is high enough,
        Then call linear regression on D and A
                 to find the slope and intercept.
             Return (A, slope, and intercept).
    Else if the maximum search depth is reached,
            Then return NIL.
    Else let C be Find-best-terms(D, S, C).
         Find-law(D, S, C).


Find-best-terms(D, S, C)

    Let P be the products of the terms of S and C.
    Let Q be the quotients of the terms of S and C.
    For each term A in the union of P and Q,
        Compute the correlation between D and A.
    Return the terms with the N highest correlations.


Parameters:
     Width of the beam (memory size).
     Threshold of the correlation (accuracy).
     Maximum power of terms (law complexity).
     Maximum depth of the search tree (when to halt).
```

fit with our description of IDS as an incremental learning system, though we hope to modify this in future versions.

Table 3 presents the basic algorithm for finding numeric laws. The top-level function `find-law` is given three arguments: the designated term `D`, the set of base terms `S`, and a set of current terms `C`. If IDS is attempting to revise an existing law, `C` is the term occurring in the right-hand side of that law. If the system is searching for a new law, `C` is the set of observable terms `S`.

At each point in the search, IDS defines the products and ratios between the terms in the set `S` and those in `C`, but it retains only those terms having the highest correlations with the designated term `D`. These new terms become the current set `C`, and the function

`find-law` is called recursively, with the designated term D and the base terms S remaining the same. If any term in C has a sufficiently high correlation with D, IDS ends the search and uses a regression technique to find the slope and intercept of the line relating them. The system continues along these lines until it finds such a linear relation or until it exceeds the maximum search depth. If the search fails, IDS assumes that no law covers all the observed data.

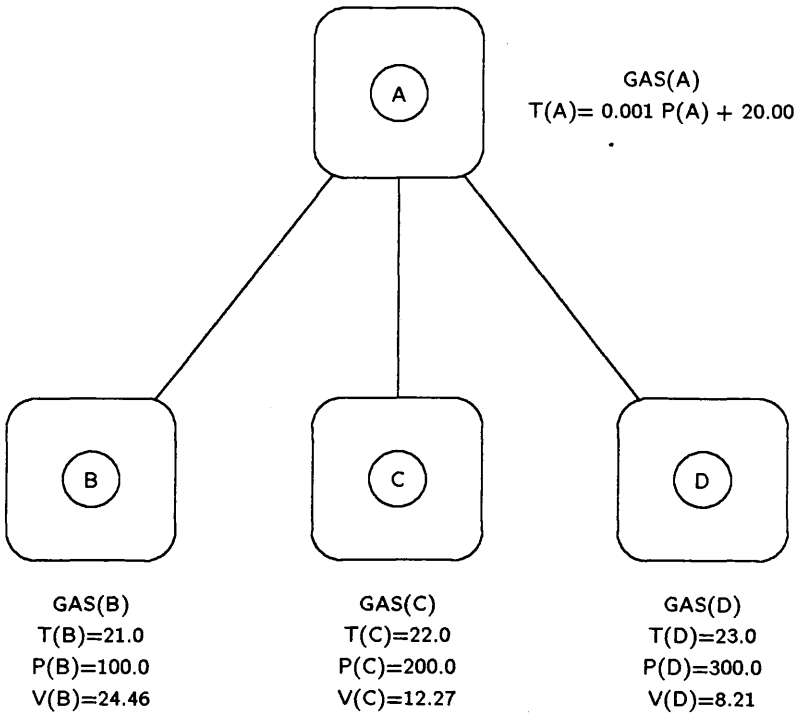

Figure 9. A spurious relation found during discovery of the ideal gas law.

As an example, let us consider how IDS rediscovers the ideal gas law. The system receives data in the form of states with gaseous objects at different temperatures, pressures, and volumes. Figure 9 shows the hierarchy after the system has processed three states, with all instances stored under a common parent node. Given these data, IDS finds a law relating the temperature and the pressure, since one can express the temperature as a linear function of the pressure. Now the system observes a fourth instance, which it adds as a child of Node 1 because it matches the parent completely.[7] However, this new instance violates the numeric law stored at the parent node, causing IDS to search for a new relation that covers all four instances.

Since the term *P* was used in the rejected law, IDS calls the function `find-law` with {P} as the current set C, T as the designated term, and {P, V, T} as the base terms S. In other words, IDS uses the term P as the entry point in the search space, starting by combining P with the terms in S to form products and ratios such as P, P^2, P/T, and P/V. Of these

[7] Recall that the system does not consider whether instances satisfy numeric laws during the clustering process.

new terms, P has a high enough correlation to end the search. Regression produces the numeric law $T = 0.12 \cdot P\ V + 273$; this version is equivalent to the standard form of the law, $P = 8.32(T - 273)$. Figure 10 shows the hierarchy that emerges after this revision is complete. As the system processes more instances and stores them under the parent node, it finds that they obey this new law, so **find-law** is not called again.

Four parameters control IDS' search for numeric laws. The size of the set of current terms C determines the beam width of the search. The level of correlation used as a termination criterion influences the accuracy of the laws and the system's tolerance of noise. Finally, the maximum power of terms and the maximum depth of the search tree limit the amount of search. Although we have not undertaken a careful study of this algorithm, preliminary results suggest that it is efficient and robust.

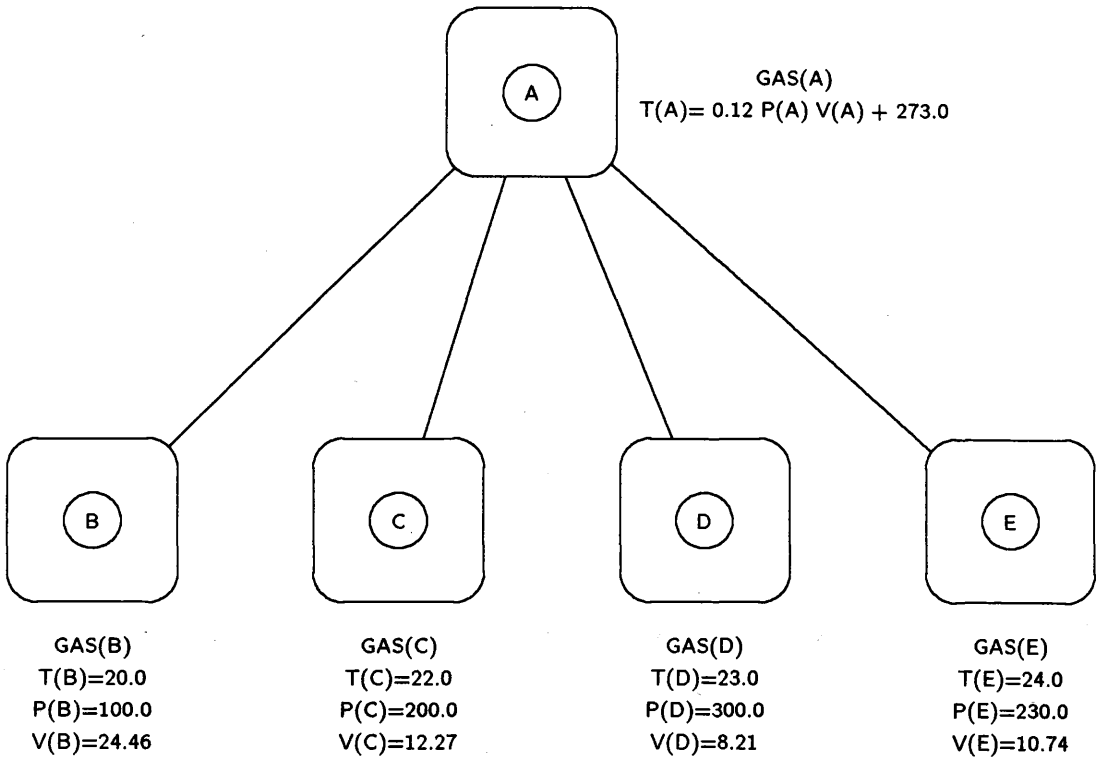

Figure 10. A correct version of the ideal gas law, found after rejection of the spurious version in Figure 9.

As noted in Section 2.2, IDS finds numeric laws similar in form to those produced by BACON (Langley et al., 1983) and ABACUS (Falkenhainer & Michalski, 1986). Moreover, they employ similar methods to control their search for useful numeric terms, using simple correlations to focus attention. However, the systems differ in the details of their search control. BACON uses a recency-based scheme, focusing on more recently-defined terms in preference to older ones. ABACUS creates a "proportionality graph" to determine promising combinations of terms, then uses a modified beam search to find the best combinations. IDS also carries out a beam search through the space of numeric terms, but this search is not as

sophisticated as that used in ABACUS. The main novelty of IDS' search scheme is the reuse of existing terms, which decreases the amount of reprocessing needed when new observations are made.

3.4 The Process of Prediction

Now that we have discussed IDS at an algorithmic level, let us consider the implications of the knowledge it acquires. The system inductively constructs a hierarchy of abstract qualitative states, augmented with qualitative and quantitative laws at different levels of abstraction. The taxonomy and its associated laws describe the observations that have been made. Embedded in the hierarchy are qualitative laws, such as the reactive behavior of acids and alkalis, and numeric laws, such as laws of combining weights, that summarize the histories given to the system.

However, these data structures also have predictive power. After IDS has observed a number of qualitative states, it can use its taxonomic hierarchy to predict unobserved states. In addition, once it has classified a novel state, it can predict the possible successors of that state (sometimes many steps ahead) and when they will occur. Finally, once the system has stored cross-state numeric laws, it can use them to predict the values of numeric attributes in as yet unobserved but predicted states. Moreover, IDS' incremental nature permits it to make these predictions at any point in the discovery process. In fact, the prediction process can be viewed as an integrated part of its discovery method.

IDS' three-tiered approach to discovery is also robust in that it can profit from partial understanding of a domain. In cases where IDS' numeric component cannot discover quantitative relations, it may still be able to form qualitative laws and use them for qualitative prediction. For example, the program can describe the qualitative behavior of reacting substances even if other factors make the combining weights difficult to determine. Similarly, in domains where temporal information is not available or not highly predictive, the system can still find within-state quantitative laws without the need to form qualitative laws. For instance, IDS needed no qualitative relations to formulate the variant of the ideal gas law shown in Figure 10. Finally, the system can construct taxonomies that organize qualitative states, and thus make simple predictions, even when no temporal information is present and no numeric laws can be found.

4. Discussion

In this section we discuss some general issues concerning our work on IDS. We begin by describing the status of the system and our ideas for near-term extensions. After this, we consider some approaches to evaluating IDS, including experiments on both historical and artificial domains. Finally, we discuss our longer-term plans for more extensive changes to the system. Although IDS integrates some important aspects of discovery, we think it holds the potential for supporting much more of the scientific process.

4.1 The Status of IDS

The three major parts of IDS – the clustering algorithm, the method for finding successor links, and the component for numeric discovery – have all been implemented. Initial experiments show that the system is able to form taxonomies and to find qualitative and quantitative laws. For instance, given histories of chemical reactions between acids and alkalis, IDS has successfully characterized the reactive behavior of these substances, as described earlier in the chapter. It has also summarized the qualitative behavior of heat exchange, along with a simple version of Black's law that does not involve specific heat. We have tested IDS on all the laws that BACON could handle which do not involve inferring intrinsic properties. The system found Ohm's law, Kepler's third law, and Coulomb's law with no difficulty.

The system has also successfully found numeric laws within states and on transition conditions. However, we have yet to implement the algorithm for inferring laws that relate attributes across states, as shown in Figure 4. We envision using a forward propagation strategy to find these relations. If no law between a state and its immediate successor can be found, the system will look for a numeric relation between the state and the successor of the successor, continuing this chain until it finds a relation or it reaches a final state. We will use the numeric discovery method from Table 3 to actually find these laws.

As Langley et al. (1983) have noted, *intrinsic properties* play an important role in empirical discovery, occurring in many numeric laws. An intrinsic property is some attribute of an object or class of objects that remains constant over time; often this attribute is not directly observable. For example, mass is an intrinsic property associated with particular objects, whereas density, specific heat, and boiling point are intrinsic properties associated with classes of objects. We are currently extending IDS to infer intrinsic properties, based on the parameters found in numeric laws within states, across states, and on transition conditions.

4.2 Evaluating IDS

Our work on integrated discovery is still in progress, and we have not yet carried out any systematic evaluation of IDS. We hypothesize that the system can discover a wide range of empirical laws, provided: (1) the qualitative states can be organized in a disjoint concept hierarchy; (2) the qualitative laws can be described as deterministic finite-state machines; and (3) the numeric relations can be stated as products of exponentiated terms. Many examples from the history of physics and chemistry satisfy these constraints, suggesting that IDS will do well in such domains. However, we need to more formally specify the space of laws searched by our algorithms, so we can identify the limits of the framework. We also need to carry out careful experimental studies of the system's behavior, along the lines proposed by Kibler and Langley (1988).

In the near future, we plan to evaluate IDS along two dimensions. As with earlier discovery systems like BACON, ABACUS, and GLAUBER, we will test IDS' ability to rediscover laws from the history of science. There are many physical and chemical relations that should fall within the system's abilities. The laws of chemical reaction that we have used throughout the chapter are obvious candidates, and we plan to borrow test cases from the earlier work on

machine discovery. However, we plan to present IDS with both qualitative and quantitative data for each of these cases. This will simultaneously test the entire system, rather than its components, in its ability to discover empirical laws of the type actually found by scientists. It may also provide more plausible historical accounts of these discoveries than earlier AI systems, although this is not our main goal.

We also plan to test IDS' ability to make predictions about unseen data. As we noted in Section 3.4, the system should be able to predict the qualitative behavior of unobserved attributes in a given state, predict the nature of succeeding states, and predict the values of numeric attributes. As IDS processes more data and its knowledge about the world improves, the accuracy of these predictions should increase. For this study we will use artificial domains, which will let us vary factors such as the structure of the taxonomy, the complexity of the qualitative laws, and the amount of noise in numeric data. We also plan to vary aspects of the system itself, such as the parameter settings used in the numeric component. Experiments of this type will provide information about the robustness of IDS' various discovery methods, and suggest ideas for improving them.

4.3 Directions for Future Research

Our long-term plans call for extending IDS in a variety of more challenging directions. These include improving the clustering method, designing experiments, and constructing new measuring instruments. We discuss each of these below.

4.3.1 Improving the Clustering Algorithm

In IDS, the discovery of qualitative and numeric laws relies upon the formation of appropriate taxonomies, making the clustering process central to the overall system. The current algorithm has some important limitations, which we hope to remedy in future work.

Our experience suggests that the clustering method is sensitive to instance order, forming different hierarchies depending on the order in which it encounters qualitative states. This feature is not so important for experimental data, since these can be presented in a careful order, with one attribute varied at a time. However, order effects can have a major impact on the structure of a taxonomy formed from observational data, and thus on the laws the system finds. Of course, any incremental hill-climbing system will have some sensitivity to order, but we would like to minimize this effect.

Fisher (1987) has argued that including additional learning operators can reduce the effect of instance order. Within the context of his COBWEB system, he describes a *split* operator that deletes a parent node and elevates its children; he also describes a *merge* operator that is similar to the one used in IDS. These give the effect of backtracking through the space of taxonomies without the need for memory of previous learning steps. Gennari, Langley, and Fisher (in press) present empirical evidence that these operators make systems like COBWEB and IDS less order dependent, and we plan to augment the IDS clustering algorithm with a split operator. This will also require the system to update its laws as the structure of the hierarchy changes.

A second drawback of the current system is that it uses a somewhat ad hoc evaluation function to sort instances and merge nodes. In future versions of IDS, we plan to adapt

Gluck and Corter's (1985) *category utility* measure, which has a theoretical grounding in information theory. Fisher's (1987) COBWEB incorporates this as an evaluation function, but his system is limited to attribute-value representations. We plan to extend the function to handle qualitative state descriptions that include multiple objects and structural relations. This approach assumes a probabilistic representation of knowledge, which should permit IDS to represent information about the likelihood of various abstractions.

Another issue involves the domain theory of substances that IDS is currently given by the programmer. This takes the form of a hierarchy, and there is no reason in principle why the system could not acquire this knowledge on its own, clustering objects with similar structural features. However, this would require IDS to construct two interleaved taxonomies, one that organizes qualitative states and another that organizes objects appearing in those states. This raises issues of updating the state hierarchy when changes occur in the object hierarchy.

A final problem concerns the assumption that the taxonomy is disjoint, with each observed state being sorted down a single path. This is a clear oversimplification for many domains, and we plan to alter the IDS clustering algorithm to form nondisjoint hierarchies, in which each node may have multiple parents. This simplest approach involves modifying the sorting process at each level of the taxonomy. In addition to considering the placement of the instance in each sibling, one can also consider placing it in the two best-matched siblings, the three best, and so forth. One then simply selects the option giving the highest evaluation score. Only experimentation will tell how well this heuristic approach works.[8]

The extension to nondisjoint taxonomies promises another benefit. Figure 11 shows a partial taxonomy in which the highest-level nodes describe simple qualitative processes, and whose children summarize states in which these processes occurred together. The extended IDS should be able to identify such primitive processes from their occurrence in more complex states, even if they are never observed in isolation. It should then be able to use these primitive processes in indexing new states that involve previously unseen combinations. For instance, suppose the system first formed Nodes 1, 2, 3, 4, and 5, and only then encountered a qualitative state of the form shown in Node 6. This new observation would be sorted through Nodes 1 and 3, and Node 6 would be stored as their joint child.

4.3.2 Experimentation

Another central component of science is experimentation, and we also hope to extend IDS to support this process. Given a current taxonomy and its associated laws, the experimentation component will design a new experiment to be run. This will take the form of an initial state, along with an (optional) set of actions by some external agent. For example, in the domain of acids and alkalis the experimenter will propose different initial states by varying the reactants, their quantities, and the initial conditions. The system will execute and monitor the experiment itself in a simulated world (Nordhausen & Langley, 1987), or ask the programmer for the results.

We plan to borrow heavily from recent work in this area by Kulkarni and Simon (in press), Rajamoney (in press), and Karp (in press). In particular, Kulkarni and Simon's

[8] We owe this idea to Doug Fisher, who originally proposed it in the context of his COBWEB system.

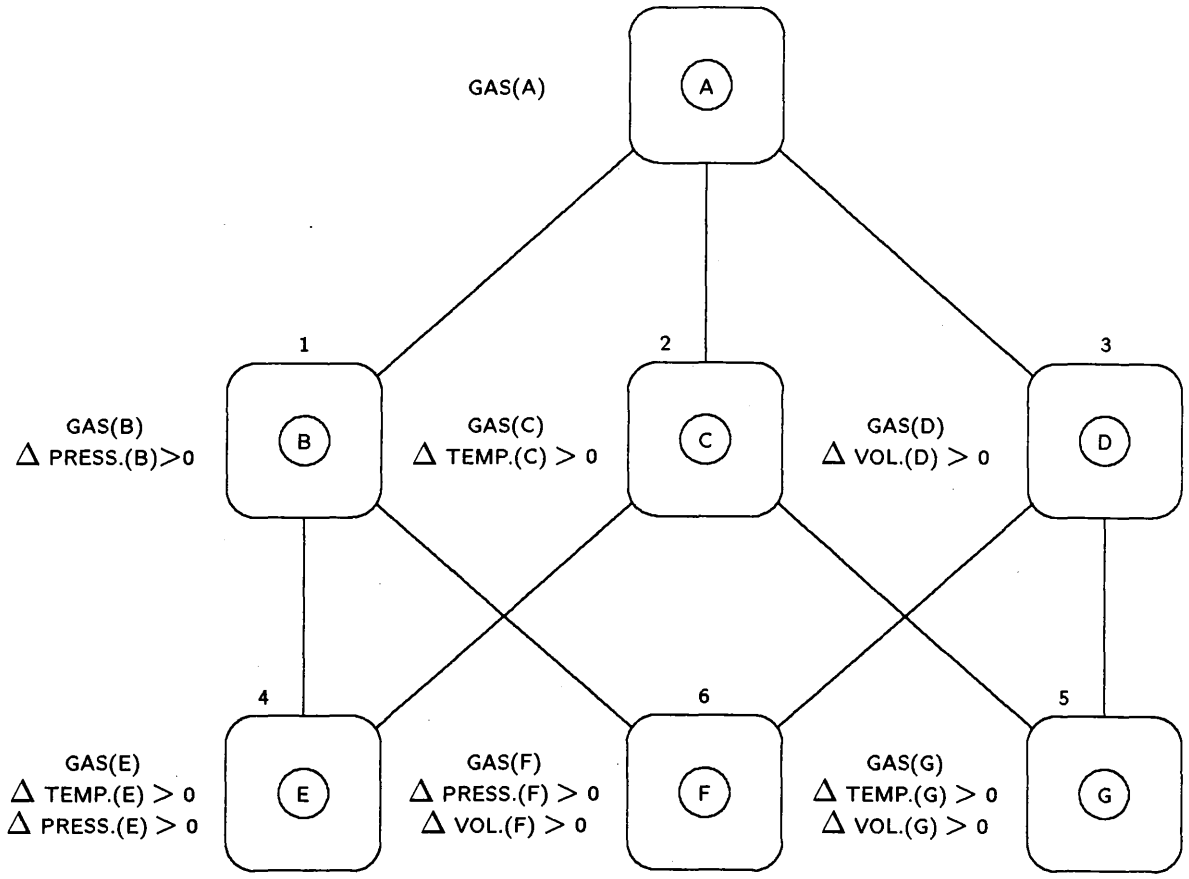

Figure 11. A nondisjoint taxonomy, with primitive processes as parents of combined processes.

KEKADA system includes a heuristic for focusing attention on surprising phenomena. In the IDS framework, one can instantiate this notion as an unpredicted qualitative state or a mispredicted numeric attribute. KEKADA attempts to identify the scope of the phenomenon, generating different initial conditions using a domain theory of substances much like the one in IDS.

This approach to experimentation will complement the data structures and mechanisms in the existing IDS system. Some of Kulkarni's heuristics, such as dropping factors that have no influence, emerge from IDS' methods for taxonomy and law formation. The incremental nature of the system will let the experimentation component change strategies after observing each history. Also, the systematic variation of substances, their relations, and their numeric attributes will simplify the clustering process, reducing the chances of undesirable order effects.

The IDS framework also supports the construction of new measuring 'instruments'. Recall that the system can store intrinsic properties, such as a substance's specific heat or boiling point. Given a sequence of abstract states containing such terms, one might place an as yet unobserved substance in the initial state and let the sequence run its course. Data observed along the way will let one estimate the specific heat or boiling point of the new

substance, effectively acting as an instrument for measuring these quantities. IDS can then use these measurement instruments in designing more sophisticated experiments.

4.4 Concluding Remarks

We have presented an integrated approach to empirical discovery that supports taxonomic hierarchies, qualitative relations, and numeric laws. Our ideas are implemented in IDS, a computational system that uses an incremental hill-climbing strategy to discover empirical laws. The system has rediscovered a number of qualitative and numeric laws from the history of physics and chemistry, and our initial experience with the system has been encouraging. However, we hope to carry out more careful experiments in the near future, using both historical and artificial domains.

The individual components of IDS borrow significantly from earlier work on empirical discovery. Each component can be improved, as we plan to do in our future work, but we believe that the overall framework is genuinely new, and that it constitutes an important contribution to our understanding of scientific discovery. Moreover, the basic framework shows the potential for covering other aspects of the scientific process, including experimentation and measurement.

Science is a complex enterprise, and it is not surprising that early work on machine discovery focused on isolated aspects of the overall process. However, the field now has relatively robust mechanisms for dealing with many components of discovery, and future progress will depend on understanding the ways in which these components interact. We think that IDS is an important step in this direction, and we encourage other researchers to join us in developing integrated frameworks for discovery.

References

Brown, J. S. (1973). Steps toward automatic theory formation. *Proceedings of the Third International Joint Conference on Artificial Intelligence* (pp. 121–129). Stanford, CA: Morgan Kaufmann.

Emde, W., Habel, C., & Rollinger, C. (1983). The discovery of the equator or concept driven learning. *Proceedings of the Eight International Joint Conference on Artificial Intelligence* (pp. 451–458). Karlsruhe, West Germany: Morgan Kaufmann.

Falkenhainer, B. C., & Michalski, R. S. (1986). Integrating quantitative and qualitative discovery: The ABACUS system. *Machine Learning, 1*, 367–422.

Fisher, D., & Langley, P. (1985). Approaches to conceptual clustering. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 691–697). Los Angeles, CA: Morgan Kaufmann.

Fisher, D. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning, 2*, 139–172.

Freund, J. E., & Walpole, R. E. (1980). *Mathematical statistics.* Englewood Cliffs, NJ: Prentice-Hall.

Forbus, K. D. (1985). Qualitative process theory. In D. G. Bobrow (Ed.), *Qualitative reasoning about physical systems.* Cambridge, MA: MIT Press.

Gennari, J. H. , Langley, P. & Fisher, D. (in press). Models of incremental concept formation. *Artificial Intelligence.*

Gluck, M., & Corter, J. (1985). Information, uncertainty and the utility of categories. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 283–287). Irvine, CA.

Hayes, P. S. (1979). The naive physics manifesto. In D. Michie (Ed.), *Expert systems in the microelectronic age.* Edinburgh: Edinburgh University Press.

Karp, R. (in press). Hypothesis formation as design. In J. Shrager & P. Langley (Eds.), *Computational models of discovery and theory formation.*

Kibler, D., & Langley, P. (1988) Machine learning as an experimental science. *Proceedings of the Third European Working Session on Learning* (pp. 81–92). Glasgow, Scotland: Pitman.

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

Kuipers, B. (1985). Commonsense reasoning about causality: Deriving behavior from structure. In D. G. Bobrow (Ed.), *Qualitative reasoning about physical systems.* Cambridge, MA: MIT Press.

Kulkarni, D., & Simon, H. (in press). Evaluation of KEKADA as an AI program. In J. Shrager & P. Langley (Eds.), *Computational models of discovery and theory formation.*

Jones, R. (1986). Generating predictions to aid the scientific discovery process. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 513–522). Philadelphia, PA: Morgan Kaufmann.

Langley, P., Gennari, J., & Iba, W. (1987). Hill climbing theories of learning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 312–323). Irvine, CA: Morgan Kaufmann.

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

Langley, P., Simon, H. A., Bradshaw, G. L., & Zytkow, J. M. (1987). *Scientific explorations of the creative process.* Cambridge, MA: MIT Press.

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

Lebowitz, M. (1987). Experiments with incremental concept formation: UNIMEM. *Machine Learning, 2,* 103–138.

Lenat, D. B. (1977). Automated theory formation in mathematics. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (pp. 833–841). Cambridge, MA: Morgan Kaufmann.

Lenat, D. B. (1982). AM: Discovery as heuristic search. In R. Davis and D. B. Lenat (Eds.), *Knowledge-based systems in artificial intelligence.* New York: McGraw-Hill.

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

Michalski, R. S., & Stepp, R. (1983). Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* San Mateo, CA: Morgan Kaufmann.

Mitchell, T. M., Utgoff, P. E., & Banerji, R. B. (1983). Learning by experimentation: Acquiring and refining problem-solving heuristics. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* San Mateo, CA: Morgan Kaufmann.

Nordhausen, B., & Langley, P. (1987). Towards an integrated discovery system. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (pp. 198–200). Milan, Italy: Morgan Kaufmann.

Rajamoney, S. (in press). A model of theory revision in scientific discovery. In J. Shrager & P. Langley (Eds.), *Computational models of discovery and theory formation.*

Thagard, P. (1988). *Computational philosophy of science.* Cambridge, MA: MIT Press.

Weld, D. (1986). The use of aggregation in qualitative simulation. *Artificial Intelligence, 30.*

Wrobel, S. (1988). Automatic representation adjustment in an observational discovery system. *Proceedings of the Third European Working Session on Learning* (pp. 253–261). Glasgow, Scotland: Pitman.

Zytkow, J. (1987). Combining many searches in the FAHRENHEIT discovery system. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 281–287). Irvine, CA: Morgan Kaufmann.