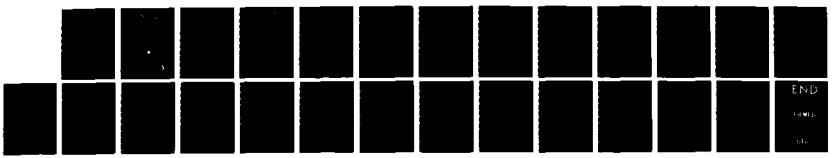
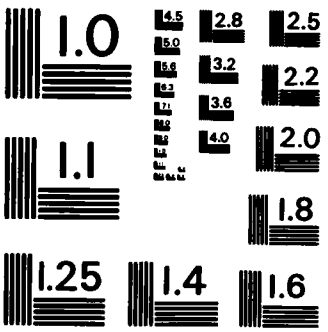
AD-A159 235 DISCOVERING QUALITATIVE EMPIRICAL LAWS(U) 1/1
CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF PSYCHOLOGY
P LANGLEY ET AL. 15 JUL 85 TR-85-18 N00014-84-K-0345
UNCLASSIFIED F/G 6/4 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

REPRODUCED AT GOVERNMENT EXPENSE

(2)

Information and Computer Science

A-1

DISCOVERING QUALITATIVE EMPIRICAL LAWS

Pat Langley[1]
Herbert A. Simon[2]
Jan M. Zytkow[3]
Douglas H. Fisher[1]

AD-A159 235

TECHNICAL REPORT

UNIVERSITY OF CALIFORNIA
IRVINE

DTIC FILE COPY

DTIC
ELECTE
SEP 13 1985
S
D

This document has been approved for public release and sale; its distribution is unlimited.

85 9 11 006

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist Avail and/or Special
A-1

DTIC COPY INSPECTED

DISCOVERING QUALITATIVE EMPIRICAL LAWS

Pat Langley[1]
Herbert A. Simon[2]
Jan M. Zytkow[3]
Douglas H. Fisher[1]

[1]Irvine Computational Intelligence Project
Department of Information & Computer Science
University of California, Irvine, California 92717

[2]Department of Psychology
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

[3]Computer Science Department
Wichita State University
Wichita, Kansas 67208

307876

Technical Report 85-18

July 15, 1985

Copyright © 1985 University of California, Irvine

This work 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 by the United States Government.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report No. 2	2. GOVT ACCESSION NO. AD-A159 135	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Discovering Qualitative Empirical Laws		5. TYPE OF REPORT & PERIOD COVERED Interim Report 1/85 - 6/85
		6. PERFORMING ORG. REPORT NUMBER 85-18
7. AUTHOR(s) Pat Langley, Herbert A. Simon, Jan M. Zytkow, Douglas H. Fisher		8. CONTRACT OR GRANT NUMBER(s) N00014-84-K-0345 Carnegie Mellon
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Information and Computer Science University of California Irvine, California 92717		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 15 July, 1985 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 P.Langley, H.A.Simon, J.M.Zytkow, & G.L.Bradshaw, Scientific Discovery: A Computational Account of the Creative Process. Cambridge, Mass.: MIT Press, 1986.

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

scientific discovery	heuristic search
qualitative laws	chemistry
conceptual clustering	theory of acids and bases

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

In this paper we describe GLAUBER, an AI system that models the scientific discovery of qualitative empirical laws. We have tested the system on data from the history of early chemistry, and it has rediscovered such concepts as *acids*, *alkalis*, and *salts*, as well as laws relating these concepts. After discussing GLAUBER we examine the program's relation to other discovery systems, particularly methods for conceptual clustering and language acquisition.

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)

Introduction

The process of scientific discovery is a complex interplay of many activities, ranging from the discovery of empirical laws through the construction of structural models to explain those laws. In an earlier paper, we forwarded the BACON system as one model for the discovery of quantitative empirical laws (Langley, Bradshaw, and Simon, 1983). Elsewhere, we have described STAHL and DALTON, two systems which address the problem of formulating structural models (Langley, Zytkow, Simon, and Bradshaw, 1983).

In this paper we will focus on the discovery of qualitative empirical laws and concepts. Our primary examples will come from the history of chemistry, and our model of the qualitative discovery process is an AI system named GLAUBER. After describing the system and providing some examples of its operation in the domain of chemistry, we will consider GLAUBER's relation to some other AI discovery systems that operate on the tasks of conceptual clustering and language acquisition. However, let us first review briefly some events from the history of science, since it was our interest in this area that led us to construct GLAUBER.

Upon examining the history of science, one finds 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 numerical relations were determined. In particular, the history of the theory of acids and bases provides us with useful insights into the discovery of qualitative concepts and 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 ammonium chloride, NH_4Cl (though the structure of this compound was of course not known). Moreover, they knew that sulfuric acid also tasted sour, and that it also combined with ammonia to form ammonium sulfate, $(NH_4)_2SO_4$. From facts like 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 our model of the qualitative discovery process.

The GLAUBER System

Our interest in the discovery of qualitative empirical laws has led us to design and implement an AI system concerned with this process. Since our main examples derive from the history of early chemistry and the theory of acids and bases, we have named the system after Johann Rudolph Glauber (1604–1670), a 17th Century German chemist who played an important role in the development of this theory. Let us begin by considering the form of data input to the system, along with the types of laws that it generates. We will then turn to the mechanisms GLAUBER uses to transform data into laws.[1] After we have described the system in the abstract, we will examine its operation on some of the data that were available to the early chemists.

GLAUBER's Representation of Data

The GLAUBER system represents data using a predicate–argument notation similar to that used in semantic networks. Each fact or observation contains a *predicate* followed by one or more labeled arguments. An example will help clarify the representational scheme.

Suppose GLAUBER[2] observes that the chemical hydrogen–chloride (HCl) reacts with ammonia (NH_3) to form ammonium chloride (NH_4Cl). This fact would be represented by the proposition (reacts inputs {HCl NH_3} outputs {NH_4Cl}). Here the predicate is *reacts*, which takes two arguments – the *inputs* and *outputs* of the reaction. GLAUBER represents the values of these attributes as sets (denoted by curly brackets), in which the order of elements is not significant. Thus, the proposition (reacts inputs {NH_3 HCl} outputs {NH_4Cl}) would be considered identical to the above fact. In our examples, we will use symbols like HCl and NH_3 for the sake of clarity. GLAUBER does not know the meaning of these symbols or the internal structure of chemicals like ammonia. Its behavior would not change is we used symbols like G00013 instead.

At first glance, GLAUBER's representation may seem identical to BACON's attribute–value scheme, save that sets can occur as values. However, note that the same symbols can occur in the arguments of other propositions, and this possibility makes for a significant difference. To see this, assume that GLAUBER inputs the fact given above, (reacts inputs {HCl NH_3} outputs {NH_4Cl}). Now suppose that the system observes a second reaction, say (reacts inputs {HCl KOH} outputs {KCl}). The occurrence of HCl in both propositions establishes a *relation* between the two facts of a sort that could not occur in a simpler attribute–value representation (eg. as in the BACON system). We will see later that GLAUBER takes advantage of such relations in its discovery process.

---

[1] The current version of GLAUBER differs from the earlier version described by Langley, Zytkow, Simon, and Bradshaw (1983). Although the state descriptions are very similar in the two systems, both the operators and the search control differ considerably.

[2] Neither the current nor the previous versions of GLAUBER perform experiments. Rather, they input a list of facts or observations provided by the programmer, and search for qualitative laws that summarize these data.

Relations (in terms of shared symbols) can also occur between facts involving different predicates. For instance, the observation that hydrogen–chloride tastes sour would be represented as (has–quality object {HCl} tastes {sour}). This fact provides another piece of information about the substance HCl that can be used in generating laws. An alternative representation of this information would use a "sour" predicate with the single argument "object".

GLAUBER's Representation of Laws

Given a set of facts, GLAUBER's goal is to find a set of laws that summarize the observed data. These laws should have the same *form* as the original facts, but specific substances should be replaced by names that denote abstract *classes* of substances, providing generality. For instance, the qualitative law (reacts inputs {acid alkali} outputs {salt}) has the same form as (reacts inputs {HCl NaOH} outputs {NaCl}), but HCl has been replaced by the class name *acid*, NaOH has been replaced by the name *alkali*, and NaCl has been replaced by *salt*. In order for such a law to have predictive power and make contact with the data, each class name must denote a non–empty list of substances. Thus, the class of acids might contain the substances HCl and HNO_3, KOH and NaOH might be alkalis, while NaCl, KCl, $NaNO_3$, and KNO_3 might be classified as salts.

However, the proposition (reacts inputs {acid alkali} outputs {salt}) contains some inherent ambiguity. Should this statement be interpreted to mean that "every acid combines with every alkali to form every salt"? Hopefully not, since this statement does not hold. In this case, we would like to say that "every acid combines with every alkali to form *some* salt" or that "every salt is a product of *some* acid and *some* alkali". These two statements are independent and complementary, and both relations generally hold (with some exceptions) for acids, alkalis, and salts. In order to distinguish between these quite different senses, we must employ some form of quantifiers.

We will use the universal quantifier $\forall$ to modify classes in which all of the members satisfy a given law, and we will use the existential quantifier $\exists$ to modify classes for which this is not the case. Thus, we can represent the statement "every acid combines with every alkali to form some salt" as "$\forall$ a $\in$ acid $\forall$ k $\in$ alkali $\exists$ s $\in$ salt (reacts inputs {a k} outputs {s})". In the examples below, we will omit the subset notation, and write simpler expressions such as "$\forall$ acid $\forall$ alkali $\exists$ salt (reacts inputs {acid alkali} outputs {salt})". Similarly, the statement that "all acids taste sour" would be represented by "$\forall$ acid (has–quality object {acid} taste {sour})".

It is important to note that the same class name may occur in different laws. Taken together, all laws that mention a given class provide an *intensional* definition of that class. This definition complements the extensional definition, since it can lead to predictions that go beyond the observed data. Also, the set of laws associated with a class is quite similar to the characterizations produced by many AI systems for learning from examples. We will return to this similarity later in the chapter. Note that current system does not have an explicit description of its goal state. Rather, GLAUBER knows it has achieved

its goal state when it generates some description that adequately summarizes the data it has observed.

GLAUBER's Discovery Method

The GLAUBER system inputs a set of observations and attempts to formulate a set of general laws that summarize these data. GLAUBER's discovery process can be usefully viewed in terms of search through a space of laws or hypotheses. Such a problem space is defined by the initial states from which search begins, by the operators used to generate new states, and by the test used to determine when the goal has been reached. We have already examined the first and last of these components, so let us now turn to GLAUBER's operators for proposing candidate laws.

In addition to the predicate and attributes that GLAUBER's laws share with the facts on which they are based, these laws involve two additional structures – the abstract classes referred to in each law, and the quantifiers placed on each class in each law. Not surprisingly, GLAUBER employs one operator for defining classes and a second operator for proposing quantifiers. We will call the first of these the FORM–CLASS operator, and the second the DETERMINE–QUANTIFIER operator. Let us consider each in turn, and then consider how they are combined to produce an effective search process.

As its name implies, the operator FORM–CLASS proposes abstract classes for use in qualitative laws. Recall that at the outset, GLAUBER has a set of propositions that vary in terms of their predicates, attributes, and values. Like most operators, FORM–CLASS can be instantiated in many different ways. In this case, each instantiation corresponds to a different combination of predicate, attribute, and value, and leads to different potential classes. For instance, based on the fact (reacts inputs {HCl NaOH} outputs {NaCl}) described earlier, it would propose three separate *sets* of classes. The first instantiation is based on the triple (reacts, inputs, HCl) and would propose one class corresponding to the second input and another corresponding to the output. Another instantiation of the FORM–CLASS operator is based on the triple (reacts, inputs, NaOH), while a third is based on the triple (reacts, outputs, NaCl).

Each such triple can be used to define one or more extensional classes based on the facts in which that triple occurs. For example, suppose GLAUBER observes the following reactions:

(reacts inputs {HCl NaOH} outputs {NaCl})
(reacts inputs {HCl KOH} outputs {KCl})
(reacts inputs {HNO_3 NaOH} outputs {$NaNO_3$})
(reacts inputs {HNO_3 KOH} outputs {KNO_3})

Given these data, the triple (reacts, inputs, HCl) defines two classes, A = {NaOH, KOH} and B = {NaCl, KCl}, while the triple (reacts, inputs, NaOH) defines two different classes, C = {HCl, HNO_3} and D = {NaCl, $NaNO_3$}. Analogous classes (each with two elements) are defined by the triples (reacts, inputs, HNO_3) and (reacts, inputs, KOH). In contrast,

the triple (reacts, outputs, NaCl) defines two single–element classes, E = {HCl} and F = {NaOH}, since the substance NaCl occurs as the output of the reacts predicate in only one fact. The three other triples involving the output attribute also define classes containing one element.

When GLAUBER is presented with a set of facts, its first step is to form tentative classes based on all observed triples, in the manner just described. Some of these classes are based on many observations, while others are based on only one or a few. GLAUBER selects the instantiation (triple) of FORM–CLASS that covers the most data, and retains the classes associated with this choice for further processing. The system also substitutes the names of these classes into propositions containing members of those classes; this leads to a smaller set of more abstract propositions. For instance, if the triple (reacts, inputs, HCl) were selected for the above data, two abstract propositions would result – (reacts inputs {HCl A} outputs {B}) and (reacts inputs {HNO_3 A} outputs {B}). Note that although the classes A and B were based on facts involving the substance HCl, the substitution process leads to their inclusion in facts involving the substance HNO_3. Thus, after the FORM–CLASS operator has been applied, GLAUBER has not only a set of initial abstract classes; it also has a set of propositions that refer to those classes. We may view these abstract propositions as candidate laws or patterns.

However, in their current form these patterns do not include quantifiers, and this is the role of the operator DETERMINE–QUANTIFIER. This operator iterates through the newly generated propositions, determining whether each class mentioned in a pattern should be existentially or universally quantified. If a single class was introduced, then this class is universally quantified in the proposition on which this class was based. In this case, the level of quantification is not an issue, since this is tautologically determined by the manner in which the classes were defined.

However, if N classes are introduced, then N instantiations of the pattern 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 pattern would be formulated – $\forall$ A ? B (reacts inputs {A NaOH} outputs {B}) and $\forall$ B ? A (reacts inputs {A NaOH} outputs {B}). The first of these states that all members of class A react with at least one member of the class B; the second states that all members of class B can be formed by at least one member of A 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–CLASS operator generates additional patterns by substituting class names for substances in other facts. In these cases, all of the quantifiers must be tested against observations. For example, the pattern (has–quality object {A} taste {sour}) might hold for all members of A, 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 pattern must be considered. Thus, if a law were formed by substituting both

A and B 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–CLASS and DETERMINE–QUANTIFIER operators, it has a revised set of facts and laws to which these operators can be applied recursively. The FORM–CLASS operator may apply to laws as well as to facts, provided these laws have identical quantifiers.[3] For example, given the two laws ∀ A ∃ B (reacts inputs {A NaOH} outputs {B}) and ∀ A ∃ C (reacts inputs {A KOH} outputs {C}), this operator would generate the more abstract law ∀ A∃ D (reacts inputs {A E} outputs {D}). In addition, it would define the class E to have the members NaOH and KOH, and define the class D with the classes B and C 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.

This process can be viewed as a form of *hill–climbing* through the space of possible laws and classes. At each point in the search, GLAUBER applies all instantiations of the appropriate operator and selects the best result. Thus, the system carries out a one–step "look–ahead" to determine the best course of action. GLAUBER's search control does not include backup capability, since its evaluation functions are sufficiently powerful to direct search down acceptable paths. Although hill–climbing methods are susceptible to local maxima, we have not encountered problems of this sort in our runs with chemical data.

To summarize, GLAUBER determines which classes to define by considering all known substances and classes, and selecting that (predicate, attribute, value) triple occurring in the largest number of facts or laws. Thus, if two facts having the predicate *reacts* and the symbol NaOH in the *inputs* slot, the triple (reacts, inputs, NaOH) would receive a score of two. 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–CLASS operator to those facts containing the highest scoring value, 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 comply with 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

[3] For the sake of consistency, one might view all initial facts as universally quantified. Thus, the proposition (reacts inputs {HCl NaOH} outputs {NaCl}) could be rewritten as ∀ HCl ∀ NaOH ∀ NaCl (reacts inputs {HCl NaOH} outputs {NaCl}).

over a class; this percentage is specified by the ser. The program interprets missing facts as unobserved; the current system cannot handle disconfirming evidence, such as ¬ ∃ salt (reacts inputs {HCl HNO_3} outputs {salt}).

TABLE 1

States generated by GLAUBER in discovering acids and alkalis

Initial state S1:

(reacts inputs {HCl NaOH} outputs {NaCl})
(reacts inputs {HCl KOH} outputs {KCl})
(reacts inputs {HNO_3 NaOH} outputs {$NaNO_3$})
(reacts inputs {HNO_3 KOH} outputs {KNO_3})
(has–quality object {HCl} taste {sour})
(has–quality object {HNO_3} taste {sour})
(has–quality object {NaCl} taste {salty})
(has–quality object {KCl} taste {salty})
(has–quality object {$NaNO_3$} taste {salty})
(has–quality object {KNO_3} taste {salty})
(has–quality object {NaOH} taste {bitter})
(has–quality object {KOH} taste {bitter})

FORM–CLASS and DETERMINE–QUANTIFIER lead to state S3:

salts: {NaCl, KCl, $NaNO_3$, KNO_3}
∃ salt (reacts inputs {HCl NaOH} outputs {salt})
∃ salt (reacts inputs {HCl KOH} outputs {salt})
∃ salt (reacts inputs {HNO_3 NaOH} outputs {salt})
∃ salt (reacts inputs {HNO_3 KOH} outputs {salt})
∀ salt (has–quality object {salt} taste {salty})
(has–quality object {HCl} taste {sour})
(has–quality object {HNO_3} taste {sour})
(has–quality object {NaOH} taste {bitter})
(has–quality object {KOH} taste {bitter})

FORM–CLASS and DETERMINE–QUANTIFIER lead to state S5:

salts: {NaCl, KCl, $NaNO_3$, KNO_3}
acids: {HCl, HNO_3}
∀ acid ∃ salt (reacts inputs {acid NaOH} outputs {salt})
∀ acid ∃ salt (reacts inputs {acid KOH} outputs {salt})
∀ salt (has–quality object {salt} taste {salty})
∀ acid (has–quality object {acid} taste {sour})
(has–quality object {NaOH} taste {bitter})
(has–quality object {KOH} taste {bitter})

FORM–CLASS and DETERMINE–QUANTIFIER lead to final state S7:

salts: {NaCl, KCl, $NaNO_3$, KNO_3}
acids: {HCl, HNO_3}
alkalis: {NaOH, KOH}

∀ alkali ∀ acid ∃ salt (reacts inputs {acid alkali} outputs {salt})
∀ salt (has–quality object {salt} taste {salty})
∀ acid (has–quality object {acid} taste {sour})
∀ alkali (has–quality object {alkali} taste {bitter})

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 1, and are very similar to facts known by 17th Century chemists before they formulated the theory of acids and bases.[4] As in our earlier examples, they consist of information about the tastes of substances and the reactions in which substances take part. 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 various (predicate, attribute, value) triples, and determining which of these occurs in the greatest number of 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*.[5] This triple is selected, and 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.

In addition to proposing this law, the FORM-CLASS operator generates four additional patterns 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 (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, (has-quality object {salt} taste {salty}), is guaranteed to be universally quantified by the manner in which the salt class was defined, the generality of the other laws must be empirically determined. For example, if the law (reacts inputs

---

[4] One might question whether these are facts or rather summaries of yet lower level observations, such as the reactions and tastes of particular objects. Indeed, one could present GLAUBER with such lower level data, and hope it would form classes corresponding to substances like HCl and KOH. The presence of additional features such as color and weight would surely aid this process. Although we have not tested this prediction, we believe that given such information, GLAUBER would be able to generate the "data" in Table 1 from lower level observations.

[5] Note that had we represented taste information using *predicates* like sour, bitter, and salty, GLAUBER would not have formulated this class. Since the system's heuristics look for shared *values*, substances' tastes must be stored as values instead of predicates.

{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 includes 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 triple occurs in the greatest number of propositions. 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 this class is existentially quantified in each of the laws, and so is not considered. Since all of the viable options involve two laws (each based on one fact), 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–CLASS 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 patterns 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 its new 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 which are based on two facts apiece, giving acid a score of four.

As a result, the two laws are passed to the FORM–CLASS operator and a higher level pattern – (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 pattern – (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 general set of laws that summarize the original data. The final laws are shown in the fourth section of Table 1, 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. As in the example above, the subclass of alkalis is identified by its taste, while the subclass of metals is set apart by its shiny color. Other than these differences, the overall discovery process is very similar to that described for the simpler task.

Limitations of the System

In its present form, GLAUBER has some important limitations, and these should be remedied in future versions of the system. The first problem revolves around the program's treatment of quantifiers and their order. Readers with background in logic will recall that $\forall$ x $\exists$ y P(x, y) is not equivalent to $\exists$ y $\forall$ x P(x, y); the second formula is more specific than the first, and thus makes stronger claims. Although GLAUBER can arrive at laws of the second form, this occurs only if it happens to define classes in a certain order; the system does not find maximally specific laws in all cases that it should.

For instance, consider the third stage in Table 1, in which GLAUBER defined the class "acid" and formulated the tautological law $\forall$ acid $\exists$ salt (reacts inputs {acid NaOH} outputs {salt}). Although this "law" was guaranteed to hold by the manner in which acids were defined, it was possible that an even stronger law held. This would occur if the *same* salt had been the output for every acid–NaOH reaction that had been observed, and could have been represented as $\exists$ salt $\forall$ acid (reacts inputs {acid NaOH} outputs {salt}). In fact, this more specific law did not describe the data, but one can imagine such cases and future versions of GLAUBER should be able to handle them.

A second problem related to the order of quantifiers involves complementary laws, such as $\forall$ x $\exists$ y P(x, y) and $\forall$ y $\exists$ x P(x, y). We have seen that such laws are considered when two classes are defined in the same step, but there are other cases in which one would like this to occur. For example, Table 1 summarizes GLAUBER's discovery of the law $\forall$ alkali $\forall$ acid $\exists$ salt (reacts inputs {acid alkali} outputs {salt}). This states that all acids and alkalis react to form some salt. However, the original data also support the complementary law $\forall$ salt $\exists$ alkali $\exists$ acid (reacts inputs {acid alkali} outputs {salt}), which states that all salts are the product of a reaction between some acid and some alkali. GLAUBER could have generated this law had its heuristics taken it down an alternative path (first defining acids, then alkalis, and finally salts), but the existing version could never generate both laws in the same run.

Another difficulty relates to the system's evaluation function for directing the search through the space of classes and laws. The current version iterates through the set of (predicate, attribute, value) triples, and selects that triple which occurs in the greatest number of 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 law that predicts a few observed reactions 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 way to implement this scheme would have GLAUBER generate the potential classes and their associated laws, in order to determine their predictive power and potential. The system would then have to consider whether these laws should be existentially or universally quantified in order to maximize their scores. In other words, the system would have to apply the FORM–CLASS 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. This is equivalent to doing a two–step look–ahead in the search tree, and 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.

However, in order to implement this strategy, we would first have to deal with two other limitations of the current system. The distinction between predictive power and predictive potential makes sense only if one can test predictions, and the testing of predictions makes sense only if such predictions can fail. This means that GLAUBER must be able to represent negated facts or "failed" observations. For instance, if the substance KOH were added to the alkali class based solely on its taste, we might predict that KOH would react with every acid to produce some salt. This abstract prediction can be stated $\forall$ acid $\exists$ salt (reacts inputs {acid KOH} outputs {salt}).

If we also know that the substances HCl and HNO_3 are acids, then two more specific predictions can be made – $\exists$ salt (reacts inputs {HCl KOH} outputs {salt}) and $\exists$ salt (reacts inputs {HNO_3 KOH} outputs {salt}). These predictions can be tested by combining the pairs of chemicals, seeing if they react, and seeing whether the output satisfies the definition of a salt. If the substances fail to react, we must represent this information in some format that GLAUBER can use, such as $\neg$ $\exists$ substance (reacts inputs {HCl KOH} outputs {substance}), or (reacts inputs {HCl KOH} outputs { }). The exact representation matters little, as long as GLAUBER knows how to interpret it. Nearly any representation is preferable to the current scheme, in which the system cannot distinguish between failed reactions and those which has simply not been observed.

In addition, GLAUBER must be able to design and run simple experiments, and this in turn requires the system to distinguish between independent and dependent terms.

For each predicate, GLAUBER must know the attributes (and values) over which it has experimental control, and which attributes it can only observe. In the case of the *reacts* predicate, one has control over all values of the *inputs* attribute, while the values of the *outputs* attribute can only be observed. For the *has-quality* predicate, one has control over the *object* being tasted, but the resulting *taste* can only be noted. Assuming such knowledge, one can easily imagine GLAUBER generating a simple factorial design (similar to that used by BACON), and using it to gather an initial set of observations. However, after some tentative classes and laws had been formulated, these could be used to generate predictions, and these in turn could lead directly to new experiments. Depending on the results on these forays, some laws might be rejected in favor of others, which would lead to yet other predictions and experiments.

This proposal suggests still other modifications to GLAUBER. The current implementation assumes that all data are present at the outset, and the system puts these data to good use in directing search through the space of laws. However, the existing version of GLAUBER is unable to respond to new data, even if these disconfirm the hypotheses it has formed. Given a data-gathering scheme like that just described, a revised version of the system might employ more *incremental* discovery methods. This might operate in the following fashion.

The revised GLAUBER would begin by selecting some predicate that involves only one independent term, such as the *has-quality* predicate, and apply this to various substances. Based on the resulting observations, the system would define initial classes and form some tentative laws, such as ∀ acid (has-quality object {acid} taste {sour}). Since only one predicate has been observed, the initial classes will have only one associated law. After this trial period, GLAUBER can run experiments using different predicates such as *reacts*, in the hope that its initial classes will lead it to further regularities.

Based on its initial classes, the system could form a number of experimental *templates*, such as (reacts inputs {acid acid} outputs {?}), (reacts inputs {acid alkali} outputs {?}), and (reacts inputs {acid salt} outputs {?}), as well as others. Each of these can be instantiated to produce specific experimental combinations, and the results can be examined. In many cases, no reaction will occur and the responsible template will be abandoned after a few instances. In this case, only one template leads to interesting results – not only do acids combine with alkalis, but the generated substance usually seems to be a salt. This law would thus be added to the intensional definition for each of the classes involved. As more data are gathered, GLAUBER may find substances that combine with alkalis to form salts, but which have no sour taste. If this occurs often enough, the reacts law may become more central to the definition of acids than the law involving taste. This would seem to be a more plausible account of the actual historical development than that provided by the current version of GLAUBER.

The incremental acquisition of data would require yet another revision – it would force us to replace GLAUBER's simple hill-climbing strategy with a more robust search method. At any given point, one set of classes and laws may best summarize the data

that have been observed. However, as predictions are tested and sometimes disconfirmed, and as new (possibly unexpected) observations are made, the current hypotheses may become untenable and alternative accounts may become preferable. Future incarnations of GLAUBER should employ a version of the best–first search, in which old options are retained for expansion as new evidence becomes available.

GLAUBER's Relation to Other Discovery Systems

Before concluding, we should spend some time examining GLAUBER's relation to other machine learning and discovery systems. Such an analysis will serve two related functions. First, it will help identify the location of GLAUBER's discovery *task* within the space of learning tasks that have been studied, and second, it will clarify the location of GLAUBER's *methods* within the space of learning and discovery techniques. We will start by addressing the first of these two issues.

The Task of Conceptual Clustering

Within the machine learning literature, researchers have identified a variety of distinct learning tasks. These include learning from examples, language acquisition, learning search heuristics, and conceptual clustering. Langley and Carbonell (1984) provide an overview of these learning tasks and the relation between them. In the following pages we will focus on the task of conceptual clustering, since this comes closest to the discovery task confronting the GLAUBER system.

Michalski (1980) is responsible for the term "conceptual clustering", and were the first to clearly formulate the class of discovery tasks denoted by this term. They proposed the notion of conceptual clustering as an alternative to traditional statistical methods for numerical taxonomy and cluster analysis (Everitt, 1980). In both cases, one is presented with a set of objects and their associated descriptions, and the goal is to generate some taxonomic scheme that groups similar objects together. For instance, one might be given a variety of animal species, along with their measurements on various dimensions. In this case, the goal would be a hierarchical classification scheme in which species were grouped into genera, families, and the like.

In traditional approaches, the analysis would stop at this point – with groupings of the observed objects at varying levels of aggregation. However, Michalski proposed that it would also be very useful to characterize each group in terms of some general description. Moreover, traditional methods usually employed a simple distance measure (between the positions of objects in an N–dimensional space) to direct the search for groupings. Michalski suggested that if concept descriptions were constructed as well, the quality of these descriptions could be used to direct the search for a useful taxonomy.

This formulation of the conceptual clustering task suggests two distinct but related subtasks. The first of these – *aggregation* – involves grouping a set of objects into subclasses (usually disjoint). The second subtask – *characterization* – involves finding some general

description for each aggregate that covers members of that group, but does not cover members of any other group. The characterization task has been widely studied under the label of "learning from examples", and various methods for solving this problem have been explored (Winston, 1975; Hayes–Roth and McDermott, 1978; Mitchell, 1977; Anderson and Kline, 1979). In learning from examples, the aggregation problem is made trivial, since instances or objects are classified by a tutor. Thus, the conceptual clustering problem can be viewed as a more difficult version of learning from examples, in which one must solve the aggregation problem in addition to finding an adequate characterization.

The conceptual clustering task differs from the task of learning from examples along another dimension as well. In the latter, only one *level* of concepts or descriptions must be discovered, while in conceptual clustering, a hierarchy of such descriptions must be generated. This introduces a whole new level along which methods for conceptual clustering may vary, as we will see when we compare some existing systems. Issues also arise about the interaction between submethods for aggregation, characterization, and hierarchy construction. Now that we have considered conceptual clustering and its position in the space of learning tasks, let us examine some specific AI systems that address this problem.

Methods for Conceptual Clustering

Mitchell (1982) has argued that learning methods can be usefully analyzed in terms of the search methods they employ, and we will follow his advice in our discussion of conceptual clustering systems. In each case, we describe the system in the abstract, and then restate its approach as search. Since we have identified three major subtasks, we consider each systems' response to the search inherent in aggregation, characterization, and hierarchy construction.

Michalski and Stepp's CLUSTER/2 (1983a, 1983b) is by far the best known conceptual clustering program. This system constructs its taxonomic hierarchy from the top down, finding aggregations and characterizations at each level. Given a set of objects, CLUSTER/2 first randomly selected N objects as *seeds* around which to "grow" clusters of objects. To this end, it employed a general–to–specific characterization technique that, for each seed object, found some description that covered that object but no other seed. Other non–seed objects covered by the description were placed in the same class as the seed object. However, the process did not stop here. CLUSTER next selected a new seed object[6] from each of the groups, and repeated the process, finding a new description for each seed, and possibly reassigning some of the objects to new groups. This continued until the seed objects stabilized, giving an optimal set of disjoint classes. At this point, CLUSTER/2 used a specific–to–general characterization technique, which produced a more conservative description than the method used on seed objects.

[6] If the quality of the clusters (in terms of their descriptions) improved over the previous round, this object was picked from the "center" of the group; otherwise, it was picked from the "edge" in an attempt to .

The system repeated this process for different values of N, giving alternative partitions of the object set, each with associated descriptions. These descriptions were used in deciding between the competitors, and the best partition was used to create the first branches in the taxonomic hierarchy. CLUSTER/2 then applied the above process recursively to each of the resulting classes, finding partitions of each set, along with their associated descriptions. Each such partition led to additional branches at lower levels of the tree, and this process of subdivision continued until further partitions ceased to provide useful summaries of the data.

Now let us reanalyze CLUSTER/2 in terms of our three levels of search, and examine the relation between these levels. The system selects an initial set of seed objects at random, but these seeds do not constitute complete aggregations. Rather, CLUSTER/2 employs a characterization method (which involves searching through a space of concept descriptions) to find some description for each seed. Each such description determines an aggregate for the seed on which it is based. However, these are not the final groupings. From each aggregate, CLUSTER/2 selects a new seed and the process is repeated, generating a new set of descriptions and a new set of aggregates.

Thus, the system uses a hill–climbing strategy in which each step involves finding an improved set of characterizations and their associated aggregates. There is a "search" for aggregations, but this is subsumed within the search for descriptions. CLUSTER/2's higher level search through the space of taxonomies is easier to follow. The system begins with a single, all–encompassing class, and successively divides this into lower level classes. However, these classes are entirely determined by the aggregation–characterization process just described, so that no additional search control is required at this level.

Langley and Sage (1984) have described DISCON, a conceptual clustering system that takes a quite different approach. The program also constructs taxonomies from the top downward, but uses knowledge of attributes and their values, rather than the more data–driven approach of Michalski and Stepp's system. DISCON carries out an exhaustive search through the space of taxonomic hierarchies, evaluating completed trees in terms of their complexity. This search process constructs an AND/OR graph, in which OR branches correspond to alternative attributes, and AND branches correspond to the values of an attribute. DISCON prefers simpler taxonomies to more complex ones that cover the same observations, and so selects that tree with the fewest number of nodes. Since the system carries out an exhaustive look–ahead, it is guaranteed to find the simplest summary of the data, though this method is expensive when many attributes are involved.

DISCON differs from Michalski and Stepp's system along a number of dimensions. In CLUSTER/2, the main search is through the space of aggregations, with a secondary search through the space of concept descriptions, the results of which are used to direct the first search. In DISCON, the main search takes place in the space of taxonomic hierarchies. At each level of the hierarchy, the systems tries to select the best description, but the quality of each description depends on the quality of the entire hierarchy. As a result, the evaluation must wait until complete trees have been constructed. Moreover,

each "description" is limited to a single attribute–value pair, rather than the arbitrary conjuncts and disjuncts of Michalski and Stepp's program.

In CLUSTER/2, the search through the space of hierarchies is degenerate, with all search occurring in the aggregation and characterization spaces. In DISCON, search through these latter spaces is degenerate, with all true search occurring in the space of hierarchies. This accounts for the vastly different "feel" one gets when reading descriptions of these systems.

In many ways, Fisher's RUMMAGE (1984) is a compromise between the two systems we have already described. Like DISCON, this program considers only descriptions that consist of single attribute–value pairs. However, rather than carrying out an exhaustive look–ahead through the space of hierarchies, RUMMAGE employs an evaluation function that requires only one–step look–aheads. Thus, at each stage in constructing its taxonomic hierarchy, the system considers all unused attributes in terms of their ability to summarize the current set of objects. For each value of an attribute, RUMMAGE constructs a description of the objects having that value. The program then computes a complexity score for all values of the attribute, and selects that attribute with the lowest score.

GLAUBER as a Conceptual Clustering System

Now that we have examined some other approaches to the conceptual clustering task, we can describe GLAUBER in the same terms. Upon reflection, we see that the system has clear responses to the problems of aggregation and characterization. The FORM–CLASS operator, and the heuristics for selecting a particular class, deals with the aggregation issue. Similarly, the DETERMINE–QUANTIFIER operator deals with characterization, along with some help from the substitution process within the FORM–CLASS mechanism. What is interesting about GLAUBER's behavior is that, unlike other conceptual clustering systems, it does not attempt to partition objects into disjoint classes all at once. In the acid–alkali example, we saw that GLAUBER first formed the class of salts and its associated laws, then found the class of acids, and only at the end did it formulate the class of alkalis. Since the system substitutes the class name for all instances of the class, it is guaranteed to find disjoint classes, but not in the traditional manner.

A second difference is that GLAUBER does not generate a complete classification for the structures it is given, which in the chemical example were reaction and taste events. Rather, it forms classes from the objects occurring in these events. Thus, GLAUBER deals with inherently relational descriptions, while CLUSTER/2, DISCON, and RUMMAGE all assume attribute–value representations. But the difference is more subtle than it may appear at first. One can imagine relational descriptions of objects, such as chairs or tables, that would still lead one to classify the objects themselves, rather than their components. The important point is that GLAUBER uses relations *between* the objects being classified in determining its taxonomic hierarchy, and this leads it to use quite different methods than other conceptual clustering systems.

Another issue relates to the direction of GLAUBER's search through the space of hierarchies. The system constructs its taxonomies from the bottom up, rather than in the divisive fashion of RUMMAGE, DISCON, and CLUSTER/2. One cannot tell this from the acid–alkali example, since it involved only two taxonomic levels – the observed substances and the abstract classes of acids, alkalis, and salts. However, the direction becomes apparent on reactions involving metals, in which the system proposes the higher level category of bases to include both metals and alkalis.

Two final differences involve the nature of GLAUBER's concept descriptions. First, the intensional definitions of classes may include existential quantifiers as well as universal ones. This is possible because GLAUBER's laws can relate different classes to each other, and these relations may hold only between subsets of class members. Since other conceptual clustering systems generate descriptions of isolated objects, existential quantifiers have no role to play. Second, GLAUBER's descriptions need not be perfect. If a law holds for most members in a class, it may still be universally quantified. This allows the system's concept definitions to have a "fuzzy" quality similar to that of many real–world concepts.

In summary, while GLAUBER has many similarities to AI systems for conceptual clustering, we found that some significant differences also exist. In many ways, GLAUBER seems to solving a somewhat different discovery task than CLUSTER/2, DISCON, and RUMMAGE. Both are concerned with forming classes and descriptions for those classes, but the former involves searching for relations between objects, while the latter systems focus on isolated objects. Almost certainly, this difference arises from the sample problems from which the systems were developed. The "mainstream" conceptual clustering systems emerged in response to work in numerical taxonomy, which was created to deal with biological data. In contrast, we developed GLAUBER in order to understand the mechanisms of discovery in early chemistry. Whether the two approaches can be combined to produce a more robust discovery method is an interesting question for future research.

Some Other Discovery Systems

As we have seen, GLAUBER can be viewed as a conceptual clustering system, but its discovery task differs somewhat from the standard definition of the conceptual clustering problem. Before closing our survey, we should briefly consider some other AI systems that are not usually viewed as conceptual clustering programs, yet which have much in common with GLAUBER. A number of these systems operate in the domain of language acquisition.

One of the most interesting (though perhaps the least known) of these systems is Wolff's SNPR (1982). In implementing this system, Wolff has explored an approach to grammar learning that incorporates methods very similar to those used in GLAUBER. SNPR 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 "thedogchased–thecatthecatchasedthedog ...", 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".[7] The main difference between the two systems' use of this operator 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 arguments, which is 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.

In our review of conceptual clustering, we divided the concept learning task into two components – a process of aggregation and a process of characterization. However, we failed to distinguish between two quite different notions of aggregation. In the first form of aggregation, one must determine which objects or events should be grouped together as *instances* of a single concept or class. This is the aggregation problem addressed by conceptual clustering systems such as CLUSTER/2, DISCON, and RUMMAGE, as well as GLAUBER. In the second form of aggregation, one must determine which objects or events should be grouped together as *parts* of a higher level object or event. Both problems are trivialized in the task of learning from examples, since the tutor groups objects into classes and specifies the parts of each object. Traditional approaches to conceptual clustering deal with *instance* aggregation, but ignore *part* aggregation.

Thus, an obvious extension ot GLAUBER would let the system form conjunctive classes or chunks, in addition to the disjunctive classes it already forms. 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,

[7] 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. Also note that, like GLAUBER, the SNPR system operates in a bottom–up fashion, rather than the top–down manner used in most conceptual clustering systems.

they produced both green and yellow children. We can represent this with propositions like (parent of {pea–2} is {pea–1}), (has–quality object {pea–1} color {green}), and (has–quality object {pea–2} color {yellow}).

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. We can accomplish this by first defining the higher level predicate *child–has–quality*, and define this chunk by the rule (child–has–quality parent {X} child {Y} color {Z}) ⇒ (parent of {Y} is {X}) & (has–quality object {Y} color {Z}). Given such a predicate, GLAUBER could rewrite its direct observations at a higher level of aggregation and form disjunctive classes based on the resulting propositions.

As a result, the system would be able to formulate laws such as ∀ pure–green (child–has–quality parent {pure–green} child {pure–green} color {green}). This states that all members of the "pure–green" class have children that are also members of that class, and that these children are green in color. This is equivalent to stating that pure–strain green peas always breed true with respect to color. Note that we have not suggested heuristics for directing GLAUBER's search through the space of conjunctive classes. Wolff's system employed a data–intensive method similar to our technique for selecting disjunctive classes. Such a method might work for an extended nonincremental version of GLAUBER, but it would not be useful for the incremental version outlined in the previous section.

Two other AI language learning systems formed both disjunctive and conjunctive classes like those generated by SNPR – Siklóssy's ZBIE (1968) and Anderson's LAS (1977). However, both systems assumed that word chunks were already known, and that the learner could tell where sentences began and ended, while Wolff's system induced both of these. In addition, both ZBIE and LAS assumed that each sample sentence was accompanied by its *meaning*, and that the goal of the learning system was to acquire some mapping between sentences and their meanings. Siklóssy represented meaning using a propositional notation, while Anderson used semantic networks, but both used this information to greatly constrain the learning process.

ZBIE and LAS employed a method for forming disjunctive classes that is a mixture of the methods used by SNPR and GLAUBER. Suppose the word X precedes the word Z in one sentence, and the word Y precedes Z in another sentence. Siklóssy's and Anderson's systems would consider creating a disjunctive class at this point, but would not follow through before examining the meaning of each sentence. Assume X', Y', and Z' stand for the concepts associated with the words X, Y, and Z. The systems would create the class (or add a word to it, if it already existed) only if X' and Y' occurred in the same relation to Z' in the meanings of the two sentences.

Thus, ZBIE and LAS required converging evidence from two sources – sequential linguistic information and relational semantic information – before forming a disjunctive class like "subject" or "verb". In contrast, SNPR relied on only the first form of information, while GLAUBER uses only the second. Although from our description ZBIE and LAS sound very similar, they actually differ in many ways, including their representation of

grammar and their solutions to the part aggregation problem. However, we have focused here on their method for handling instance aggregation, since this holds the most relevance to GLAUBER.

GLAUBER also bears some resemblance to Brown's (1973) discovery system, which operated in the domain of kinship relations. This system noted relations that held empirically between predicates in its data base. For instance, it might discover that whenever the relation brother (X, Y) holds, the relations parent (Z, X) and parent (Z, Y) also hold. Such relations are actually more like relational versions of Wolff's sequential chunks than GLAUBER's disjunctive classes, which Brown's program did not define. Also, Brown's system focused on finding redundancies between facts in a data base, rather than creating higher level terms to summarize a set of observations. Some more recent work by Emde, Habel, and Rollinger (1983) addresses a problem very similar to Brown's task. In this case, the method examines whether predicates obey certain higher–level relations, such as transitivity or inversivity. Although this approach leads to laws very similar to those found by Brown, their model–driven discovery method contrasts with the data–driven technique used in the earlier system.

Conclusions

To summarize, our interest in the discovery of qualitative empirical laws led us to design and implement GLAUBER, an AI system that operates in this domain. Given a set of observations, GLAUBER defines abstract classes and formulates laws stated in terms of these classes. Our approach was driven by examples from the history of early chemistry, specifically by the development of the theory of acids and bases. Although the existing version of GLAUBER covers many of these discoveries, it has numerous limitations that should be remedied in future versions of the system. These include the need for improved evaluation methods, the ability to distinguish between unobserved and unsuccessful reactions, and the ability to run simple experiments in order to test predictions. These improvements suggest the need for two additional revisions – methods for the incremental discovery of classes and laws, and a search organization more robust than the current hill–climbing scheme.

Despite GLAUBER's limitations, its relations to other AI discovery systems are interesting in their own right. We found that GLAUBER has much in common with conceptual clustering systems such as Michalski and Stepp's CLUSTER/2, but we found significant differences as well. These included differences in the representation of data and laws, and in the details of search through the space of laws and classes. GLAUBER is also closely related to AI language acquisition systems, in particular to Wolff's SNPR. In this case the differences between the systems suggested another extension to GLAUBER – the inclusion of an operator that forms conjunctive classes or chunks, to let the system restate observations at higher levels of aggregation.

As usual, more work remains to be done, and we intend to implement a revised version of GLAUBER that incorporates many of the extensions we have outlined. However, the

current instantiation of the system has already provided us with an interesting account of the qualitative discovery process, and it has led to a variety of intriguing questions that we plan to pursue in our future research.

References

Anderson, J. R. (1977). Induction of augmented transition networks. *Cognitive Science, 1*, 125–157.

Anderson, J. R. and Kline, P. (1979). A learning system and its psychological implications. *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, 16–21.

Brown, J. S. (1973). Steps toward automatic theory formation. *Proceedings of the Third International Joint Conference on Artificial Intelligence*, 20–23.

Emde, W., Habel, C., and Rollinger, C. (1983). The discovery of the equator or concept driven learning. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, 455–458.

Everitt, B. (1980). *Cluster Analysis*, Heinemann Educational Books, Ltd.

Fisher, D. (1984). *A Hierarchical Conceptual Clustering Algorithm.* Technical Report, Department of Information and Computer Science, University of California, Irvine, CA.

Hayes–Roth, F. and McDermott, J. (1978). An interference matching technique for inducing abstractions. *Communications of the ACM, 21*, 401–410.

Langley, P., Zytkow, J. M., Simon, H. A., and Bradshaw, G. L. (1983). Mechanisms for qualitative and quantitative discovery. *Proceedings of the International Machine Learning Workshop*, University of Illinois, Urbana–Champaign, 121–132.

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

Langley, P., Zytkow, J. M., Simon, H. A., and Bradshaw, G. L. (1986). The search for regularity. To appear in R. S. Michalski, J. G. Carbonell, and T. M. Mitcheli (Eds.), *Machine Learning: Volume 2.* Palo Alto, Ca.: Morgan–Kaufman Publishers.

Langley, P. and Sage, S. (1984). Conceptual clustering as discrimination learning. *Proceedings of the Fifth Biennial Conference of the Canadian Society for Computational Studies of Intelligence.* 95–98

Langley, P. and Carbonell, J. G. (1984). Approaches to machine learning. *Journal of the American Society of Information Science, 35*, 306–316.

Michalski, R. S. (1980). Knowledge acquisition through conceptual clustering: A theoretical framework and algorithm for partitioning data into conjunctive concepts. *Interna-*

tional Journal of Policy Analysis and Information Systems, 4, 219–243.

Michalski, R. S. and Stepp, R. E. (1983a). Automated construction of classifications: Conceptual clustering versus numerical taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5, 396–409.

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

Mitchell, T. M. (1977). Version spaces: A candidate elimination approach to rule learning. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 305–310.

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

Siklóssy, L. (1972). Natural language learning by computer. In H. A. Simon & L. Siklóssy (Eds.), *Representation and Meaning: Experiments with Information Processing Systems*. Englewood Cliffs, NJ: Prentice–Hall.

Winston, P. H. (1975). Learning structural descriptions from examples. In P. H. Winston (Ed.), *The Psychology of Computer Vision*. New York: McGraw–Hill.

Wolff, J. G. (1982). Language acquisition, data compression, and generalization. *Language and Communication*, *2*, 57–89.

END

FILMED

11-85

DTIC