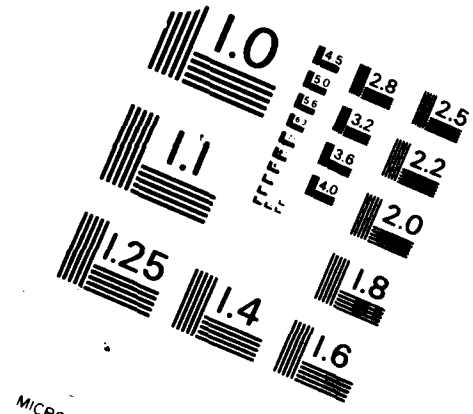
AD-A170 883 GENERATING PREDICTIONS TO AID THE SCIENTIFIC DISCOVERY PROCESS(U) CALIFORNIA UNIV IRVINE DEPT OF INFORMATION AND COMPUTER SCIENCE R JONES 15 JUL 86 1/1

UNCLASSIFIED UCI-ICS-TR-86-16 N00014-84-K-0354 F/G 9/2 NL



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

AD-A170 883

(12)

Information and Computer Science

GENERATING PREDICTIONS TO AID THE SCIENTIFIC DISCOVERY PROCESS

Randy Jones
Irvine Computational Intelligence Project
Department of Information and Computer Science
University of California, Irvine, CA 92717

TECHNICAL REPORT

DTIC
ELECTE
AUG 8 1986
S D
B

UNIVERSITY OF CALIFORNIA IRVINE

DTIC FILE COPY

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

86 8 089

GENERATING PREDICTIONS TO AID THE SCIENTIFIC DISCOVERY PROCESS

Randy Jones
Irvine Computational Intelligence Project
Department of Information and Computer Science
University of California, Irvine, CA 92717

DTIC
ELECTE
AUG 8 1986
S D
B

Technical Report 86–16

July 15, 1986

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

Copyright © 1986 University of California, Irvine

This work was supported by Contract N00014–84–K–0345 from the Information Sciences Division, Office of Naval Research.

Unclassified

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

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report No. 5	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Generating Predictions to Aid the Scientific Discovery Process		5. TYPE OF REPORT & PERIOD COVERED Interim Report 1/86–4/86
		6. PERFORMING ORG. REPORT NUMBER UCI–ICS Technical Report 86–16
7. AUTHOR*(s)* Randy Jones		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 Information Sciences Division Office of Naval Research Arlington, Virginia 22217		12. REPORT DATE July 15, 1986 13. NUMBER OF PAGES 15
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 Proceedings of the National Conference on Artificial Intelligence, 1986.

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

machine learning
conceptual clustering
qualitative empirical laws
scientific discovery
experiment generation

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

OVER

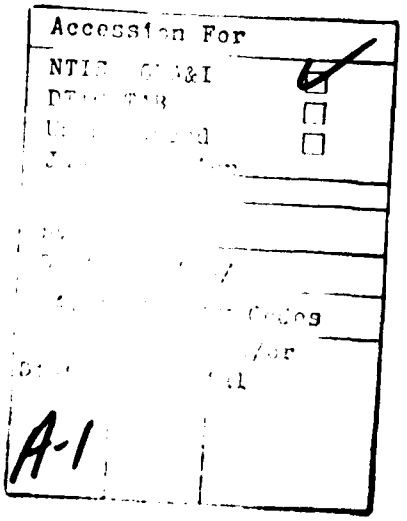
DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

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

20. ABSTRACT

NGLAUBER is a system which models the scientific discovery of qualitative empirical laws. As such, it falls into the category of scientific discovery systems. However, the program can also be viewed as a conceptual clustering system since it forms classes of objects and characterizes these classes. NGLAUBER differs from existing scientific discovery and conceptual clustering systems in a number of ways: It uses an incremental method to group objects into classes; these classes are formed based on the relationships between objects rather than just the attributes of objects; the system describes the relationships between classes rather than simply describing the classes; and most importantly, NGLAUBER proposes experiments by predicting future data. The experiments help the system guide itself through the search for regularities in the data.



Introduction

Studying scientific discovery from a machine learning point of view is still a relatively new idea. So far there have been only a few systems which attempt to model aspects of this area [5,7]. In this paper we will discuss NGLAUBER, a system which searches for regularities in scientific data and makes predictions about them. NGLAUBER is based on an earlier system called GLAUBER [6] but contains a number of differences from that system. NGLAUBER accepts its input incrementally and proposes experiments to improve its characterizations of the input. We will discuss NGLAUBER's architecture and give a simplified example of NGLAUBER at work. Finally, we will discuss NGLAUBER's relation to other systems in the area of machine learning. These include conceptual clustering systems and systems which model scientific discovery.

Data representation in NGLAUBER

To begin our discussion of the NGLAUBER system we will describe the data representation scheme. NGLAUBER deals with four basic entities. These are *facts*, *nonfacts*, *predictions* and *classes*. The two basic units of data are *objects* and *statements*. Objects are the items which are described by statements. Anything can be an object, from a block to a chemical to a qualitative description. Every statement is composed of a relation name, a set of input objects (or independent variables), and a set of output objects (or dependent variables). The general form is $relation(\{\text{Inp}_1, \ldots, \text{Inp}_m\}, \{\text{Out}_1, \ldots, \text{Out}_n\})$. For example, a statement describing the taste of the chemical NaCl would look like

$$taste(\{\text{NaCl}\}, \{\text{salty}\})$$

which simply means that NaCl tastes salty.

Statements may also be quantified over any classes that have been formed. For instance if the salts were the class of all chemicals which taste salty, then the following fact might appear in memory:

$$\forall x \in \text{salts: } taste(\{x\}, \{\text{salty}\}$$

If some — but not all — of the salts tasted salty, this statement would be existentially quantified ($\exists$) rather than universally quantified ($\forall$).

Facts, nonfacts and predictions are just sets of statements which have special meanings to NGLAUBER. A fact simply represents a statement which NGLAUBER knows is true. In contrast, a nonfact looks just like a fact, but it represents a statement which NGLAUBER knows is not true. A prediction is represented as a pair of statements (Prediction, For), where Prediction is a statement which NGLAUBER believes may be true and For is a statement which is true if the Prediction is true. An example is the prediction

$$\text{Prediction: } taste(\{\text{KCl}\}, \{\text{salty}\})$$
$$\text{For: } \forall x \in \text{salts} : taste(\{x\}, \{\text{salty}\}).$$

If NGLAUBER makes this prediction it is saying that it will know that all salts taste salty if it sees that KCl tastes salty (KCl is a member of the class of salts). The Prediction part of a prediction is always an instantiation of the For part.

Classes are sets of objects which appear as input or output values in various statements. A class is formed when a set of objects is found to have properties in common based on existing facts. The class of salts might be stored in memory as

$$\text{salts} = \{\text{NaCl, KCl}\}$$

The classes are used to allow simple statements to be rewritten as quantified statements as shown above. The exact methods for forming classes and quantifying statements will be detailed later.

The four mechanisms of NGLAUBER

NGLAUBER is an incremental discovery system with the ability to make predictions about the data it is given.[1] These two properties are natural companions for a number of reasons. It is unnecessary to make predictions with an all-at-once system because the system knows no more data is coming. The ability to make predictions is made possible by incrementality. For NGLAUBER's task, making predictions is not only desirable, but necessary. This is because when facts are quantified some information can be lost. *Predictions allow that information to be retained.* This problem will be discussed more completely later.

Langley, et al [6] describe GLAUBER as a set of operators being applied cyclically to a working memory. The same approach could be used to describe NGLAUBER but there is so much interaction between various rules that it is more convenient to divide the system into four main *mechanisms*. We will describe each of these mechanisms in turn. They are referred to as the *introduction* mechanism, the *prediction* mechanism, the *prediction satisfaction* mechanism, and the *denial* mechanism. These mechanisms can also be considered in two separate groups. The introduction, prediction, and prediction satisfaction mechanisms work together in a highly recursive manner to create classes and quantified facts. The denial mechanism works separately to prune down the number of predictions in memory and to handle the nonfacts.

The introduction mechanism

This is the main section of the NGLAUBER system. When a new fact is input to the system, the introduction mechanism decides what will happen to it. The most interesting thing that may happen is that the fact will cause a new class to be formed. NGLAUBER

[1] The GLAUBER system has neither of these properties. It can form classes and descriptive facts as NGLAUBER does, but it uses a much different method and must have all its data available at once.

does not have the advantage of knowing all the facts it is going to see when it is time to form a class.[2] However, this should not keep the system from forming classes whenever possible. Currently a simple heuristic is used to solve this problem. When two facts are identical in all but one position, a new class is formed containing the two differing objects. Using this rule, the two facts can be combined to form one universally quantified fact because they define the class at this point. Any other facts involving the objects in the class can be existentially quantified.

The introduction mechanism takes care of all of these activities. Every fact in memory (and every future fact) which involves an object in some class becomes existentially quantified. As an example, suppose NGLAUBER's memory contains the two facts

$$\textit{color}(\{\text{block1}\}, \{\text{blue}\})$$
$$\textit{shape}(\{\text{block1}\}, \{\text{cube}\})$$

and the following fact is introduced to the system:

$$\textit{color}(\{\text{block2}\}, \{\text{blue}\})$$

At this point the class {block1,block2} is formed and the facts are quantified. Now NGLAUBER's memory would look something like:

$$\text{class1} = \{\text{block1,block2}\}$$
$$\forall x \in \text{class1: } \textit{color}(\{x\}, \{\text{blue}\})$$
$$\exists x \in \text{class1: } \textit{shape}(\{x\}, \{\text{cube}\})$$

It can be seen that some information has been lost during this operation. NGLAUBER now knows that there is *some* object in class1 which is cube shaped, but has seemingly forgotten *which* object the fact is true for. This problem is conveniently taken care of when the ability to make predictions is added. This is our next topic of discussion.

The prediction mechanism

There are certain problems associated with NGLAUBER's introduction mechanism due to its incrementality. At any given point in time, the system does not know if it has seen all the data it is going to see. Therefore, it assumes that it will receive no more input when forming its classes and facts. However, it must also be flexible enough to alter its memory in a correct and appropriate manner if it does receive more input.

A desirable characteristic for such a system is to have some expectation of what it will see in the future. When possible, NGLAUBER's prediction mechanism performs this task. Predictions are made which will allow the system to easily expand its facts when the predictions are satisfied.

[2] In contrast, GLAUBER is an all-at-once system. Because of this, NGLAUBER's criterion for forming a new class is quite a bit different from GLAUBER's.

The prediction mechanism works on the assumption that every existentially quantified fact can eventually become a universally quantified fact if the proper data is seen. Referring to the example in the previous section, when *class1* is formed the following prediction is also made:

prediction: *shape*({block2}, {cube})
for: $\forall x \in$class1: *shape*($\{x\}$, {cube})

The implicit assumption in this type of prediction making is that the domain is highly regular. NGLAUBER believes that if objects have one thing in common then they will probably have many things in common. Therefore, when it sees that block1 and block2 are both blue and that block1 is a cube, it decides that block2 will probably be a cube too.

The predictions in NGLAUBER's memory are generally highly interrelated. There can be many predictions with the same *prediction* part. Likewise, there can be many predictions with the same *for* part. The set of predictions with the same *for* part is called a *prediction group.* Also, the *for* statement that is common to every prediction in a group is called the *hypothesis* of the group. Another way to think of the predictions in a prediction group is as a conjunctive implication. To know that the *for* statement in a prediction is true, it is not enough for just one prediction to be satisfied. Rather, *every* prediction in the same group must be satisfied before it is known that the *for* statement (i.e. the hypothesis) is true.

It can be seen that the predictions also conveniently solve the loss of information problem mentioned earlier. When the fact *shape*({block1}, {cube}) is quantified to $\exists x \in$ class1: *shape*($\{x\}$, {cube}), predictions are made at the same time. These predictions act as sort of a sieve. They tell NGLAUBER which statements have not yet been seen, so it also knows which statements *have* been seen. The net result is a reorganization of information with no loss. A benefit of this is that NGLAUBER will come up with the same classes and facts for a given input set regardless of the order of the input. This is a trait which is often not exhibited by incremental systems.

The prediction satisfaction mechanism

Working hand in hand with the prediction mechanism is the prediction satisfaction mechanism. The prediction satisfaction mechanism is invoked by the introduction mechanism to see if the current fact has been predicted by the system. Satisfying a prediction is usually just a matter of 'checking off' the fact from the list of predictions kept in memory. When a predicted fact is introduced to the system, all predictions of that fact are removed from memory. Often this is the only thing that happens when this mechanism is invoked.

A special case occurs when the last prediction in a prediction group is removed from memory. As explained earlier, NGLAUBER knows at this point that the hypothesis of the prediction group is true. This allows NGLAUBER to make stronger claims about the data it is considering. When this occurs, the prediction mechanism invokes the introduction

mechanism with the newly confirmed fact. This completes the recursive cycle between the first three mechanisms. When NGLAUBER introduces a new fact to itself via the prediction satisfaction mechanism, the cycle begins again. New predictions may be made or satisfied, and new classes may be formed by the introduction of the new fact.

The denial mechanism

The final mechanism to be discussed is a bit different from the previous three. It is a separate entity which cannot be invoked by the other mechanisms. Neither does it call any of them into action. The task of the denial mechanism is to correctly reshape NGLAUBER's memory when a prediction has been made which turns out to be false. This mechanism does not do anything to the facts in memory. This is because the facts only summarize everything which NGLAUBER knows to be true. To deny something NGLAUBER knows to be a fact would mean that the data is noisy. Currently NGLAUBER is not designed to deal with noise so there would be unpredictable consequences.

The real effect of the denial mechanism is to prune down the number of predictions in memory. We saw earlier that all the predictions in a prediction group had to be satisfied in order for the hypothesis of the group to be true. By way of the denial mechanism, we can tell NGLAUBER that one of these predictions is not true. If that is the case, then NGLAUBER knows that the hypothesis can *never* be true.

This revelation allows the denial mechanism to perform two tasks. The first is to eliminate all predictions in the same group as the denied statement. At the same time, the statement is recorded as a nonfact to keep any future prediction groups involving the statement from being formed. The reason for eliminating the predictions is not because they have been satisfied. Rather, NGLAUBER no longer *cares* whether they are true because it already knows that the hypothesis of the group is not true.

This knowledge is also the justification for the second task of the denial mechanism. Since the hypothesis of the prediction group cannot be true, it also qualifies as a nonfact. Therefore, the denial mechanism loops back, using the hypothesis as the denied statement. This can lead to more predictions being removed from memory. The cycle will continue until there are no more predictions left which can be removed. All the while, nonfacts will be recorded in memory but the classes and facts will never be touched.

An example of NGLAUBER at work

In this section we give a simplified example of NGLAUBER at work on a task. We will use the same input data as used for GLAUBER by Langley, et al [6]. The example is from the domain of eighteenth century chemistry. Given a set of reactions between elements and descriptions of the tastes of the chemicals, NGLAUBER forms the classes of acids, alkalis and salts. The system also comes up with a set of facts which describe these classes and the interactions between the classes.

Following are the data input to the system. They were entered in the order shown, but it should be reemphasized that NGLAUBER is order-independent. No matter which order the facts are input, the system will end up in the same state.

1. *reacts*({HCl,NaOH}, {NaCl})
2. *reacts*({HCl,KOH}, {KCl})
3. *reacts*({HNO_3,NaOH}, {$NaNO_3$})
4. *reacts*({HNO_3,KOH}, {KNO_3})
5. *taste*({HCl}, {sour})
6. *taste*({HNO_3}, {sour})
7. *taste*({NaCl}, {salty})
8. *taste*({KCl}, {salty})
9. *taste*({$NaNO_3$}, {salty})
10. *taste*({KNO_3}, {salty})
11. *taste*({NaOH}, {bitter})
12. *taste*({KOH}, {bitter})

The first five facts listed are just added into NGLAUBER's memory unchanged because NGLAUBER has found no reason to form a class. However, when fact number six is introduced more interesting things start to happen. To begin with, NGLAUBER notices that both HCl and HNO_3 taste sour. Using this knowledge a class containing those two objects is formed. We will refer to the class as 'acids' although NGLAUBER would use a generic name like 'class1'. The generalization process of the introduction mechanism then alters the *reacts* facts to describe the new class. For instance, facts one and three are changed to

$\exists x \in$acids : *reacts*({x,NaOH}, {NaCl})
$\exists x \in$acids : *reacts*({x,NaOH}, {$NaNO_3$})

Facts two and four are changed similarly. Notice now that NGLAUBER can form two new classes based on these new *reacts* facts. Using the new facts one and three, the class salts1 = {NaCl, $NaNO_3$} will be formed. Likewise, using facts two and four, NGLAUBER comes up with salts2 = {KCl, KNO_3}. After everything has been completed, NGLAUBER's memory will look something like this:

acids = {HCl, HNO_3}
salts1 = {NaCl, $NaNO_3$}
salts2 = {KCl, KNO_3}
$\rightarrow \forall z \in$ salts1 $\exists x \in$ acids : *reacts*({x, NaOH}, {z})
$\rightarrow \forall x \in$ acids $\exists z \in$ salts1 : *reacts*({x, NaOH}, {z})
$\forall z \in$ salts2 $\exists x \in$ acids : *reacts*({x, KOH}, {z})
$\forall x \in$ acids $\exists z \in$ salts2 : *reacts*({x, KOH}, {z})
$\forall x \in$ acids : *taste*({y}, {sour})

Now is a good time to point out that the space of quantified facts is only partially ordered. By examining the new facts marked by arrows, for example, we see two descriptions which summarize the data and yet do not subsume each other. It would be possible for one of these facts to be true without the other. This partial ordering is discussed more in the next section. NGLAUBER finds *all* the characterizations which apply to a given set of data.[3]

During this whole process predictions are being made about future data. We have omitted listing them because most of them are not true and will never be useful. On this and similar example runs, sixty to eighty–five percent of NGLAUBER's predictions turned out to be false.[4] These will later be removed with the denial mechanism. However, when fact seven is introduced, a useful prediction is made. When NGLAUBER sees that NaCl tastes salty, it predicts that $NaNO_3$ will also taste salty. The same occurs with fact eight. KNO_3 is predicted to taste salty. There is a great deal more that happens when fact number eight is introduced.

At this point, NGLAUBER has two distinct classes — we are calling them salts1 and salts2. When NGLAUBER sees that members of each class have something in common (i.e. they both taste salty), it decides that these two classes should really be one class and merges them. We will refer to this new class simply as 'salts'. A consequence of this merger is that facts currently in memory now describe only one class rather than two. This means that a new class can be formed containing NaOH and KOH. This is the class of 'alkalis'. After all appropriate quantifications have been made to the existing facts, NGLAUBER's memory contains:

- acids = {HCl, HNO_3}
- alkalis = {NaOH, KOH}
- salts = {NaCl, KCl, $NaNO_3$, KNO_3}
- $\forall x \in$ acids $\forall y \in$ alkalis $\exists z \in$ salts : $\mathit{reacts}(\{x, y\}, \{z\})$
- $\forall x \in$ acids $\exists z \in$ salts $\exists y \in$ alkalis : $\mathit{reacts}(\{x, y\}, \{z\})$
- $\forall y \in$ alkalis $\forall x \in$ acids $\exists z \in$ salts : $\mathit{reacts}(\{x, y\}, \{z\})$
- $\forall y \in$ alkalis $\exists z \in$ salts $\exists x \in$ acids : $\mathit{reacts}(\{x, y\}, \{z\})$
- $\forall z \in$ salts $\exists x \in$ acids $\exists y \in$ alkalis : $\mathit{reacts}(\{x, y\}, \{z\})$
- $\forall z \in$ salts $\exists x \in$ acids $\exists y \in$ alkalis : $\mathit{reacts}(\{x, y\}, \{z\})$
- $\forall x \in$ acids : $\mathit{taste}(\{x\}, \{\text{sour}\})$
  $\exists z \in$ salts : $\mathit{taste}(\{z\}, \{\text{salty}\})$

NGLAUBER's memory will also contain two important predictions, that $NaNO_3$ and KNO_3 taste salty. This ensures that when facts nine and ten are seen, the last fact in

[3] This is something which GLAUBER does not do. It stops when it has found *one* of the characterizations which apply.

[4] Of course it is possible to tailor examples where all of the predictions are false or none of them are.

NGLAUBER's memory will be changed to

- $\forall z \in$ salts : *taste*($\{z\}, \{\text{salty}\}$)

Facts eleven and twelve now simply result in the fact

- $\forall y \in$ alkalis : *taste*($\{y\}, \{\text{bitter}\}$)

being added to memory. The only job left is to get rid of all the useless predictions lying around. By denying all the false predictions NGLAUBER has made, such as

$$\textit{reacts}(\{HNO_3, NaOH\}, \{NaCl\})$$

NGLAUBER's final contents will consist only of the classes and quantified facts that are marked with bullets (•). These final quantified facts represent the relationship between the classes of acids, alkalis, and salts that was discovered in the eighteenth century.

NGLAUBER as a conceptual clustering system

In this section, we will examine the NGLAUBER system using Fisher and Langley's framework for conceptual clustering algorithms [2,3]. This framework includes three classes of techniques used in conceptual clustering and divides the conceptual clustering task into two main problems. The three types of techniques are:

1. *Optimization* — Partitioning the object set into disjoint clusters,
2. *Hierarchical* — Creating a tree, where each leaf is an individual object and each internal node is a cluster and
3. *Clumping* — Creating independent clusters which may overlap.

The two problems of conceptual clustering are defined as:

1. *Aggregation* — The problem of deciding which objects will be in which clusters and
2. *Characterization* — The problem of describing the clusters once they have been formed.

NGLAUBER uses an *optimization* technique because its classes are simply partitions of the set of objects. The classes are disjoint, but they cover all the objects. Actually, it is possible for objects to end up unclassified but each of these can be considered as a class of one object.

The *aggregation problem* is solved for NGLAUBER by the heuristic used for forming classes. As stated previously, classes are formed when two facts are found to differ in exactly one position. This problem has actually become simpler because of the incrementality of the system. When a new fact is input, it only has to be compared to the existing facts in memory in order to possibly form a new class.

The new class is then *characterized* by the quantification process which changes facts describing objects into facts describing classes. This problem is also relatively simple since the initial facts are used as templates to form the new facts.

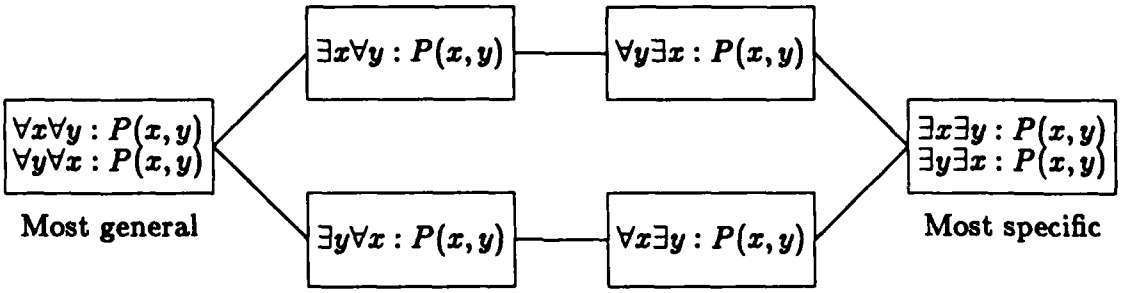
An important difference in characterization from other systems is in the quantified facts which describe the classes. Existing conceptual clustering systems form clusters and come up with *one* defining characterization for each cluster [1,8]. In contrast, there is usually not just one fact which defines a class in NGLAUBER (or GLAUBER). More often, there is a set of facts involving a class which describes its relationships with other classes. The reason this occurs is that classes are formed and described using the *relationships* between objects. In other systems, clusters are formed strictly by examining the *attributes* of each object.

This type of description requires the use of existential quantifiers. However, existential quantifiers are desirable because they increase the power of the description language. Without them, facts like

$$\forall x \in \text{acids}\ \forall y \in \text{alkalis}\ \exists z \in \text{salts} : \textit{reacts}(\{x, y\}, \{z\})$$

are not possible. Most existing conceptual clustering systems would have trouble generating this type of description.

This brings us to the discussion of NGLAUBER's *characterization space.* As mentioned previously, the concept descriptions used by NGLAUBER are partially ordered with respect to generality. For this reason there is usually more than one applicable characterization for a given set of data. Consider statements which have two quantifiers. We can draw a direct analogy to mathematical logic with predicates of two variables. Following is a diagram of the partial ordering involving a predicate $P(x, y)$ from general to specific, where the truth of more general statements imply the truth of more specific statements.



This same ordering holds on the characterization space of NGLAUBER. When more than one characterization applies to a set of data given to NGLAUBER, it will generate *every* maximally general quantified description which is true.

NGLAUBER as a discovery system

The GLAUBER system was designed to model the discovery of qualitative empirical laws. This is just one important aspect of the general field of scientific discovery [5,6]. Since NGLAUBER is based on GLAUBER, it is meant to address and expand on these same issues.

NGLAUBER examines a set of scientific data and attempts to characterize the regularities occuring within the data. This is considered to be an important first step in the scientific discovery process. One can envision NGLAUBER as part of a larger discovery system. NGLAUBER's task might be to search for qualitative regularities and prompt another system, such as BACON [4], to do a more in-depth quantitative analysis.

The main improvement of NGLAUBER over GLAUBER is its ability to make predictions. When NGLAUBER makes a prediction, it is effectively proposing an experiment to be carried out and asking for the results. By proposing experiments, the system is telling the user what it thinks is interesting and should be looked at more closely. It is obviously desirable for a discovery system to guide its own search for regularities. The prediction mechanism of NGLAUBER is a step in that direction. Most current discovery systems (and conceptual clustering systems) are completely passive. They simply characterize data without attempting to report which data would be more helpful to know about.

A notable exception to this rule is Lenat's AM [7]. AM not only proposes experiments in arithmetic but carries them out itself. AM also searches for regularities among data to form special classes. In theory, AM could come up with the same classes as NGLAUBER does but it would complete this task in a very different manner. The philosophy in AM is to explore a concept space looking for 'interesting' things. However, unless the interestingness functions built in to AM were highly specific, it seems unlikely that the concepts discovered by NGLAUBER would be discovered by AM in a short amount of time (if ever). The main difference between the systems is that NGLAUBER has a well-defined goal to attain. It is attempting to change a set of input facts which describe objects into a set of maximally general quantified facts which describe classes of objects. In contrast, AM has no specific state it is trying to reach. It just performs a search through the space of possible concepts led by its interest functions. This works wonderfully in the domain of pure mathematics, but does not seem easily transferable to more applied domains.

Summary

We have examined a system called NGLAUBER. Although NGLAUBER was originally designed as a scientific discovery system, it can also be viewed as a conceptual clustering system. It should be clear that at least part of scientific discovery involves searching for regularities in data and creating clusters based upon these regularities.

NGLAUBER's main contributions involve its incremental nature. Previous discovery systems need all their data at the outset and perform all-at-once computations. In contrast, NGLAUBER examines its data a piece at a time, allowing it to be more flexible in its characterizations of the data. Incrementality also allows NGLAUBER to interact with the user by making predictions or proposing experiments about the data it has seen so far. In this way, the system can guide itself through the data space until the proper characterizations are found.

In the field of conceptual clustering, incrementality is also seldom used. As stated above, NGLAUBER's characterizations are more flexible to change as more data comes in. This may lead to non-optimal classes in some cases but the trade-off is the ability to make predictions about future data. NGLAUBER also bases its classes (or clusters) on relational information rather than information about the attributes of objects. This is something that has not been seen in other conceptual clustering systems. Finally, NGLAUBER has a more powerful description language through the use of existential quantifiers. This allows the system to describe relations between classes rather than just giving definitions for each class separately.

Future work

There are many directions in which this work can be extended. NGLAUBER is an important first step toward discovery systems which design their own experiments. However, to become really useful it must be made more sophisticated in some areas. One needed improvement is in the heuristic used to form classes. This rule is simple and cheap since it allows NGLAUBER to complete its task using no search (and therefore no backtracking). However, the rule is also rather naïve. A more sophisticated version of NGLAUBER might form classes from facts which differ in more than one position. In this case, a number of hypotheses for the "best" classes (according to some evaluation function) would be remembered. Unfortunately, this method would also require search.

Viewing NGLAUBER's classes as clusters and the quantified facts as characterizations we can consider NGLAUBER to be a conceptual clustering system. Using this knowledge, we should be able to look to the conceptual clustering literature for possible extensions to NGLAUBER. Another important improvement would be to incorporate a hierarchical technique or perhaps a clumping technique for clustering rather than the current optimization technique. Arranging the classes as a tree would allow more flexible clusters and characterizations to be formed. This is something we hope to do in the near future. We envision a version of NGLAUBER which will be able to construct a periodic table of elements when given sets of reactions similar to those given in our example. To complete this task, NGLAUBER would need to have a class for each row of the table *and* a class for each column.

More research needs to be done in the area of prediction-making. NGLAUBER's current method simply uses the goal of changing existentially quantified facts into universally quantified facts. Although this method has turned out to be useful, more intelligent and complicated predictions could be made by adding some domain-specific knowledge to the system. Currently, NGLAUBER just looks for obvious regularities in the data and usually generates a large number of predictions. A little intelligence about the domain being examined would limit the number of predictions made and allow NGLAUBER to propose a few specific experiments to be performed.

Finally, an ideal NGLAUBER system would be able to deal with a certain amount of noise. Currently the system demands absolute regularity in the data to form classes and

universally quantified facts. A more flexible system would be able to make rules describing how *most* of the items in a class behave. This would remove the assumption that *all* items in a class have everything in common. This problem is closely tied with the problem of making more intelligent predictions. A future version of NGLAUBER might carefully select a set of experiments to perform. If most of these experiments succeed or fail then NGLAUBER can come up with a statement that is *generally* true or false. However, if some experiments succeed and some fail, it would imply that the system has an improper understanding of the true concept. In this case, NGLAUBER would design more specific experiments to come up with more refined classes and characterizations.

References

[1] Fisher, D. "A Hierarchical Conceptual Clustering Algorithm", Technical Report 85–21, Department of Information and Computer Science, University of California, Irvine, 1984.

[2] Fisher, D. and Langley, P. "Approaches to Conceptual Clustering", In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (1985), 691–697.

[3] Fisher, D. and Langley, P. "Methods of Conceptual Clustering and their Relation to Numerical Taxonomy", Technical Report 85–26, Department of Information and Computer Science, University of California, Irvine, 1985.

[4] Langley, P., Bradshaw, G. L. and Simon, H. A. "Rediscovering Chemistry with the BACON System", In *Machine Learning: An Artificial Intelligence Approach*, Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (editors), Tioga Publishing Co., Palo Alto, Ca., 1983, 307–329.

[5] Langley, P., Zytkow, J. M., Simon, H. A. and Bradshaw, G. L. "The Search for Regularity: Four Aspects of Scientific Discovery", In *Machine Learning: An Artificial Intelligence Approach, Volume 2*, Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (editors), Morgan–Kaufman Publishers, Los Altos, Ca., 1986, 425–469.

[6] Langley, P., Zytkow, J. M., Simon, H. A. and Fisher, D. H. "Discovering Qualitative Empirical Laws", Technical Report 85–18, Department of Information and Computer Science, University of California, Irvine, 1985.

[7] Lenat, D. B. "Automated Theory Formation in Mathematics", In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (1977), 833–842.

[8] Michalski, R. S. and Stepp, R. E. "Learning from Observation: Conceptual Clustering", In *Machine Learning: An Artificial Intelligence Approach*, Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (editors), Tioga Publishing Co., Palo Alto, Ca., 1983, 331–363.

END

DTIC

9-86