The Knowledge Engineering Review, Vol 6:4, 1991, 259–305

Computational models of scientific discovery

SAKIR KOCABAS

Marmara Scientific and Technological Research Centre, PK 21, Gebze, Kocaeli, Turkey

Abstract

Computational modelling of scientific discovery is emerging as an important research field in artificial intelligence. Various computational systems modelling different aspects of scientific research and discovery have been developed. This paper looks at some of these models in order to examine how knowledge is organized in such systems, what forms of representation they have, how their methods of learning and representation are integrated, and the effects of representation on learning. The paper also describes the achievements and shortcomings of these systems, and discusses the obstacles in developing more comprehensive models.

1 Introduction

In the last ten years, computational modelling of discovery has become an increasingly active field of research in artificial intelligence. A number of researchers have developed various computational systems modelling different aspects of scientific discovery. Some provide detailed accounts of certain discoveries in the history of science; others focus on the psychological validity of a computational model; still others are concerned with finding algorithms that can be used in scientific discovery (Shrager & Langley, 1990). Computational study of science differs from conventional philosophical studies both in its aims and methods.

Shrager and Langley (1990) point out the basic differences between the philosophical and computational approaches to science as follows: conventional philosophical tradition emphasizes the evaluation of laws and theories, while the computational approach emphasizes scientific discovery, including the activities of theory formation, law induction and experimentation. They also argue that the earlier philosophical approaches focus on the structure of scientific knowledge, whereas the computational approach focuses on the processes of scientific thought and on describing these activities in computational terms.

Modern scientific research is one of the most complex human activities, involving the formulation of research goals, choosing goals, strategies and methods, experiment design, conducting experiments, data collection, data evaluation, hypothesis formation and theory revision. Each of these activities can involve dealing with different types of problem states. For example, data evaluation involves dealing with erroneous, spurious, inconsistent and incomplete data in an efficient way. Therefore, scientific research usually requires a great deal of background knowledge and planning, even though the degree of complexity may change from one field of research to another. Therefore, any formal analysis of scientific research is bound to be incomplete. Nevertheless such an analysis is necessary for computational studies of science.

The motivations behind the computational studies of science can be outlined as follows:

1 To explore the types of knowledge and skills necessary for scientific discovery in various fields of science.
2 To explore the methods and techniques of qualitative and quantitative learning and discovery.
3 Computational investigation of the historical development of ideas in a theoretical field.
4 Logico-mathematical analysis of domain knowledge by axiomatization.

5 Theoretical analysis of domain knowledge.
6 Theory development.
7 Building intelligent personal research assistants.

Let us now examine how far we stand from achieving these goals. First, how much do we understand about the knowledge and skills necessary for scientific research?

1.1 Types of knowledge and skills

In an attempt to build an operational concept system for the computational studies of science, Shrager and Langley (1990) describe scientific behaviour in terms of two basic components: knowledge structures, which includes the descriptions of observations, taxonomies, laws, theories, background knowledge, models, explanations, predictions and postdictions, and anomalies; and processes, which include a series of scientific activities such as observation, taxonomy formation, inductive law formation, theory formation, deductive law formation, explanation, prediction, experimental design, manupulation, evaluation and theory revision. Their classification is not aimed at distinguishing the types of knowledge necessary for scientific research and discovery per se, but to provide a detailed and useful terminology for computational studies of science. Therefore, a classification of the types of knowledge and skills is needed.

Knowledge necessary for scientific research and discovery can be loosely divided into four different types:

1 Commonsense knowledge.
2 Technical knowledge.
3 Theoretical knowledge.
4 Methodological knowledge.

Commonsense knowledge is simple, general and relatively unstructured knowledge about the world. Statements such as 'Fire burns', 'Water is fluid' can be cited as examples at commonsense knowledge. Lenat and Feigenbaum (1987) emphasize the importance of representing commonsense knowledge as well as theoretical knowledge in intelligent systems. In fact, Lenat's CYC project (Lenat & Guha, 1989) relies on commonsense knowledge in overcoming the problem of brittleness in such systems.

Technical knowledge can be defined as knowledge about instruments, methods and processes. Knowledge about how to repair a TV set, how to control a chemical reactor and how to fly an aeroplane can be considered as technical knowledge. Theoretical background is helpful, but not usually essential, in acquiring this kind of knowledge. On the other hand, technical knowledge helps acquiring the technical skills necessary for scientific research through practice.

Theoretical knowledge is structured knowledge about the world, which embodies classifications and numerous interrelated hypotheses or principles. Typical examples of theoretical knowledge are the classical mechanics and electrodynamics. Theoretical knowledge may sometimes run contrary to commonsense knowledge and intuition. Such knowledge is expressed in many forms, including compound propositions such as "if-then" rules (or rules of inference).

Methodological knowledge, on the other hand, is mostly characterised by "if-do" rules (i.e. rules of action), and roughly corresponds to what is called "heuristics" or "compiled hindsight" by Lenat and Feigenbaum (1987). It includes knowledge about how to distinguish between scientifically interesting and uninteresting phenomena, how to choose between alternative strategies, methods and processes in scientific research, how to design experiments, how to propose new hypotheses, and how to generalize, test and evaluate (e.g. verify or falsify) them. It is mostly the extent of this type of knowledge that determines the difference between a research scientist and a nonscientist.

Unlike the inference rules in theoretical knowledge, many of the methodological rules (e.g. rules of hypothesis formation) rely on extra-logical methods (inductive generalization, abduction, abstration and analogy). These rules are frequently used in covering large search spaces during the

activity of scientific research. Detailed models of scientific research and discovery must extensively use methodological knowledge besides commonsense, technical and theoretical knowledge.

Most of these four types of knowledge are believed to be computationally representable, as long as they are propositionally expressible. On the other hand, as Shrager and Langley (1990) and Tweney (1990) emphasize, there are some other kinds of knowledge that are not so easily representable in current systems. First of all, visual information in the forms of pictures, diagrams, tables, charts, etc., which are so frequently used by humans is one example. Other sensory information can also be added to the list. However, Tweney (1990) reminds the representational power of mathematics in summarizing many physical forms of 'imagery' and its use in modern science (e.g., in the theoretical physics).

1.2 Methods of representation, learning and discovery

One aim for the computational study of science is to identify the methods of qualitative and quantitative learning and discovery employed in scientific research, and to clarify and formulate them for use in computational models. Since such systems are basically knowledge systems, the ways of representing knowledge must also be identified. With the emergence of the first computational models, there has been a steady development in the exploration of knowledge representation, learning and discovery methods. The integration of different knowledge representation methods such as rules, predicate statements, frames and schemata in these systems has been accomplished. However, the integration of different methods of learning and discovery has proved to be a more difficult task, and only a few systems have some degree of such integration.

A variety of learning methods and strategies have been used in computational systems. Lenat's (1979) AM uses general heuristics in constructing new and useful concepts from a set of domain concepts. EURISKO (Lenat, 1983 b) also employs a set of general rules for rule discovery. Both systems represent their formal knowledge in frames. Langley's (1978) BACON system operates with four general heuristics for discovering the quantitative relationships between the variables of physical systems. Langley et al.'s GLAUBER (1987) uses a form of conceptual clustering (Michalski & Stepp, (1983) in modelling the early chemical discoveries, Zytkow and Simon's (1986) STAHL uses domain heuristics in building the componential models of chemical substances. Rose and Langley's REVOLVER (1988) employs algebraic and domain constraints in its theory revision. Kilkarni and Simon's (1988; 1990) KEKADA uses general heuristics in directing its research tasks and proposes experiments, which form the basis of its rediscoveries.

Nordhausen and Langley's (1987) IDS integrates the methods of qualitative and quantitative discovery. IDS was one of the earliest discovery systems which implemented qualitative process schemas (Forbus, 1984) for representing qualitative changes in physical systems. BR-3 (Kocabas, 1991 a) uses several completeness and consistency constraints in directing its theory formation and revision operations. The system increases its domain knowledge by a combination of abstraction and abduction. BR-3 also uses algebraic constraints in search control during its theory revision.

AbE (O'Rorke et al. 1990), COAST (Rajamoney, 1990) and GEN-SIM/HYPGENE (Karp, 1990) rely on incremental changes in their process schemas for their theory revision. PI (Thagard & Holyoak, 1985) uses analogy, based on structural similarity and conceptual combination in its discoveries. GALILEO (Zytkow, 1990), on the other hand, relies on matching process descriptions with algebraically transformed expressions of quantitative laws in discovering more general forms of quantitative laws.

1.3 Computational study of historical development

Computational studies of science can also help to better understand the history, logic and psychology of scientific discovery and development, and contribute to the development of the history and philosophy of science. To a certain extent, computational models such as STAHLp (Rose & Langley, 1986), AbE (O'Rorke et al, 1990), and ECHO (Thagard & Novak 1990) help us

in understanding the steps that may have been involved in paradigm shifts[1] and scientific revolutions such as the transition from the phlogiston theory to the oxygen theory in 18th century chemistry, and the geological revolution in the 20th century. Even the earlier systems like GLAUBER (Langley et al., 1987) and STAHL (Zytkow & Simon, 1986) tell us something about the early historical development of modern chemistry. These programs enable us to computationally test whether our historical explanations of such scientific developments are logically and formally consistent. On the other hand, quantitative models of discovery such as BACON (Langley et al., 1987) and GALILEO (Zytkow, 1990) can be helpful in tracing the developments that led to more complex laws (e.g. Ideal Gas Laws and Kirchhoff's Law) from the expansion or combinations of simple laws (e.g. Boyle's Law, Ohm's Law).

1.4 Logico-mathematical analysis

The design and development of computational models of scientific discovery require logical, and often also mathematical, analysis of the domain knowledge on which these models operate. Inconsistent domain knowledge can lead a computational model into inconsistent results, which can sometimes be more easily detectable than the contradictions in the original knowledge, as has been observed in the behaviour of BR-3 (Kocabas, 1991 a).

Logico-mathematical analysis and transformations that have been applied by GALILEO (Zytkow, 1990) can be another example to see how scientists manipulate the mathematical expressions of certain scientific laws, and find more useful expressions of these laws in creating partial models of more complex physical systems.

1.5 Theoretical analysis

Theoretical analysis is a step that usually succeeds logico-mathematical analysis. In theoretical analysis, qualitative and quantitative relationships between theoretical domain concepts are investigated, and empirical and theoretical relationships are identified. An important but less recognized feature of BACON (Langley et al., 1987) is that this program can identify the intrinsic properties of substances of physical systems as it attempts to discover the mathematical relationships between the variables. Studies on the intrinsic properties of substances in the past had lead the physicists to the investigation of the structure of matter, and eventually to the discoveries of more fundamental aspects of physics and chemistry such as the modern atomic and molecular theory.

Among the computational models developed to date, PI (Thagard, 1988), ECHO (Thagard & Novak, 1990), and more significantly GALILEO (Zytkow, 1990) provide examples of conceptual and theoretical analysis and synthesis. The first system simulates the discovery of the wave theory of sound, while the second one models the gradual transition from the earlier theories to modern geological theory through a series of conceptual changes. The third system finds the more generally applicable expressions of quantitative laws, and is interesting in the way it performs its analysis and synthesis of mathematically represented theoretical expressions or terms.

Theoretical analysis through conceptual changes is an important part of scientific development, and more computational models are expected to emerge in the future to investigate other revolutionary theoretical discoveries such as Maxwell's equations, special theory of relativity, and quantum theory. These discoveries involve conceptual analysis as well as theoretical analysis and synthesis, while the current systems such as PI, ECHO and GALILEO perform only one of these activities.

[1] Paradigm shifts are transitions from one theory to another one that has more explanatory power in the same domain (see Kuhn, 1970).

1.6 Theory development

Theory development involves theory formation, theory revision and paradigm shifts (see Rajamoney, 1990). Theory formation involves generating hypotheses by various logical and extralogical methods over empirical data and less general hypotheses. Logical methods include deductive generalizations such as explanation-based generalization (Mitchell et al., 1986). On the other hand, extralogical methods include abstraction, abduction, perturbations and analogy.

In theory revision, hypotheses that are inconsistent with observations, or between themselves, are revised, or entirely new hypotheses are generated and tested. Theory revision is carried out by using domain-specific consistency constraints as in STAHL; by using algebraic constraints as in REVOLVER and BR-3; or by experimentation as in KEKADA, COAST and GENSIM/HYPGENE, which compare their experiment results with their hypotheses.

Modelling paradigm shifts was first tackled by STAHLp. This program provides a partial model of the transition from the phlogiston theory to the oxygen theory in 18th century chemistry. The AbE system, which was developed later, provides a more detailed account of the same process, using qualitative schemas. Kulkarni and Simon's (1988, 1990) KEKADA uses general heuristics in directing its research tasks, and proposes experiments which form the basis of its theory development. Nordhausen and Langley's (1987) IDS integrates the methods of qualitative and quantitative discovery. IDS was one of the earliest discovery systems which implemented qualitative process schemas (Forbus, 1984) for representing qualitative changes in physical systems.

BR-3 (Kocabas, 1991 a) uses algebraic rules for search control in its theory revision, and increases its domain knowledge by a combination of abstraction and abduction. O'Rorke et al.'s (1990) AbE combines theory revision and paradigm shifts. This system's basic method of discovery is abductive inference on its qualitative process descriptions. Rajamoney's (1990) COAST and Karp's (1990) GENSIM/HYPGENE rely on incremental changes over their process schemas for their theory revision.

Computational study of theory development also provides tools for working in artificial domains. For this reason, domain data is sometimes changed arbitrarily to see what kind of theoretical and conceptual alterations these changes would bring. This enables the computational scientist to define new relationships and design new 'thought experiments'. This, in turn, can help increase the knowledge of the field in an indirect way. Scientists can be too busy in trying to build ever more comprehensive theories of the physical reality, and as a result, less interested in 'what if' scenarios. Whereas, such scenarios can help to better understand why the current theory is the best one (at least for the benefit of the students of science), and can be much more easy to generate with the computational models of the domain.

1.7 Building artificial research assistants

Another motivation behind the computational studies of science is ultimately to build artificial research assistants. This has been explicitly stated first by Buchanan & Feigenbaum (1978), and later by several other researchers, e.g., Michalski (1986), and Kulkarni and Simon (1988). DENDRAL (Buchanan & Feigenbaum, 1978) was one of the earliest computational systems successfully used in scientific research. Initially it operated like an expert system, applying a set of heuristics in a narrow domain. Later, it was supported with another module, Meta-DENDRAL (Buchanan & Feigenbaum, 1978), which was capable of formulating heuristics for the DENDRAL system itself. Despite this advance, the whole system remained capable of performing only a limited part (data evaluation and hypothesis formation) of a series of activities that form scientific research. Similarly, MOLGEN (Friedland, 1979) was confined to planning experiments in molecular biology.

Lenat's (1979) AM was capable of rediscovering a number of mathematical concepts, and several conjectures from a collection of set theoretic concepts and operations. Mathematical and

formal discovery does not involve most of the activities involved in empirical research such as experimental design, data collection and evaluation, hypothesis formation and testing and theory revision. Other computational models involved in empirical discovery, represent only limited aspects of empirical research. Therefore, a comprehensive computational model must integrate all the representable elements of scientific research.

Current systems are far from this integration. However, there have been some significant attempts in this direction, such as Nordhausen and Langley's (1987) IDS, which integrates data collection, qualitative and quantitative modeling, and theory formation. Unlike other empirical discovery systems, IDS also has some degree of autonomy. KEKADA (Kulkarni & Simon, 1988) constitutes another attempt towards integrating various research activities such as choosing research problems, strategies and methods, proposing and designing experiments, data collection, hypothesis formation and testing, and theory revision. However, it must be noted that some of KEKADA's activities are simulated by the user, and the system lacks the limited autonomy possessed by IDS.

In summary, we are still a long way from developing artificial research assistants due to a series of limitations, some of which can be overcome in the medium term. Some of these limitations are expressed by Tweney (1990): modelling higher order heuristics; analogical interaction between seemingly unrelated domains; representing visual imagery; maintaining and using a large knowledge base; and data evaluation.

Table 1 summarizes the developments in the computational modelling of discovery in the last 12 years. The table indicates improvements in the knowledge representation in such systems. Early systems used frames (AM, EURISKO), lists (BACON, GLAUBER, STAHL), and categorized predicate statements (BR-3). These models were restricted to representing episodic changes in physical systems. Later, with the help of qualitatitive process schemas, more detailed models were developed (IDS, KEKADA, COAST and GENSIM/HYPGENE).

There has also been some development in the representation of prescriptive knowledge. Early models such as AM used rules as demons attached to the slots of concept frames. This representation facilitates search control, but restricts autonomy in learning search control. Other systems such as BACON, GLAUBER, STAHL, KEKADA and BR-3 separate their descriptive and prescriptive knowledge, and represent the latter as a set of rules or procedures and operators. KEKADA constitutes an advance in this direction, as this system's prescriptive knowledge is organized into a set of research operators, each consisting of a set of condition-action rules. More recently, Kocabas (1991 b) described how prescriptive knowledge can be organized into a hierarchy of such research operators so that a system can learn its control rules by various methods of learning, such as explanation-based generalization (Mitchell et al., 1986), and classification (Nilsson, 1965).

As for the developments in search methods used in these models, almost all known computational search methods (depth-first, breadth-first, heuristic, best-first and assumption-based methods) have been employed, in some systems two or more in combination. The first two methods are exhaustive search methods, and can only be used in small search spaces. Heuristic search can be general or domain restricted, depending on the generality of the constraints, while best-first search relies on an evaluation function and an agenda mechanism. Some systems such as REVOLVER, BR-3 and GALILEO employ algebraic constraints. The use of such constraints can be expected to expand in future systems.

Some earlier computational models such as BACON, GLAUBER and STAHL can be regarded as data driven systems, while KEKADA, AbE, COAST and GENSIM/HYPGENE are theory driven systems; yet some others (AM, EURISKO, IDS, BR-3 and GALILEO) start as data driven systems and gradually acquire theory driven characteristics as they can use their accumulated knowledge in directing their search. The success of data driven models like BACON may suggest that large amounts of knowledge are not a prerequisite for scientific discovery. However, this would be misleading, because even the most successful data driven models of discovery are far from being able to represent a number of research activities involved in scientific discovery. These

Table 1 Comparisons of various computational models of discovery

Discovery system	*Approximate date of design*	*Research goals*	*Descriptive knowledge representation*	*Prescriptive knowledge representation*	*Search methods/ strategies*	*Search constraints*	*Theory revision methods*
AM	1977	mathematical discovery	frames	rules (demons)	best-first search	heuristics	none
BACON	1978	quantitative discovery	lists	rules, procedures	heuristic search	general heuristics	
EURISKO	1983	formal discovery	frames	rules (demons)	best-first search	general heuristics	none
GLAUBER	1983	empirical discovery	lists	rules	heuristic, depth-first	domain heuristics	none
STAHL	1986	empirical discovery	lists	rules	heuristic search	domain heuristics	uses simple consistency constraints
IDS	1987	qualitative/quantitative discovery	qualitative schemas	rules	heuristic search	general heuristics	
KEKADA	1988	empirical discovery	process schemas	organized rules	best-first search	general/domain heuristics	experimentation
BR-3	1989	consistency/ completeness	lists	consistency & completeness operators	heuristic depth-first	algebraic, domain heuristics	incremental; uses abstraction and abduction
AbE	1990	consistency	qualitative schemas	theory revision operators	heuristic search	domain heuristics	incremental; uses abduction
COAST	1990	consistent explanations	qualitative schemas	theory revision operators	heuristic search	domain heuristics	experimentation
GENSIM/ HYPGENE	1990	consistent explanations	frames qualitative schemas	theory revision operators	best-first, assumption-based methods	domain heuristics	experimentation
GALILEO	1990	more expressive laws and models	qualitative schemas	transformation rules	heuristic, depth-first	algebraic rules	

activities include: formulating research goals, choosing a research goal among possible alternatives, setting the framework for research (e.g., for knowledge and data gathering), strategy choosing, designing experiments (and sometimes tools for experiments), data collection and evaluation, hypothesis formation, theory revision and generating exlanations.

Earlier computational models such as AM, EURISKO, BACON and GLAUBER were theory formation systems that did not have theory revision capabilities. In other words, they could not revise their knowledge when faced with contradictions with the input data. After its first but simple implementation in STAHL, theory revision has become almost a standard feature of successive systems, as can be seen in KEKADA, BR-3, AbE, COAST and GENSIM/HYPGENE. It should also be noted that, in its brief history, computational study of science has tackled the complex process of paradigm shifts in science. Thanks to the research in this direction by Rose and Langley (1986), and O'Rorke et al. (1990), we can hope to understand more clearly the steps in which such fundamental changes occur in science.

After this introduction we can now examine in more detail some of the computational models of discovery, in terms of their knowledge organization and representation, inputs and outputs, rules of inference and rules of action (heuristics), methods of learning and discovery, and their results. In order to provide a systematic review, these systems are grouped as follows: molecular structure and rule discovery systems in organic chemistry; planning and discovery in molecular biology; mathematical and formal discovery; theoretical discovery; theory revision and discovery in physics and chemistry; quantitative discovery.

The paper is organized accordingly as follows: section 2 describes DENDRAL and Meta-DENDRAL, whose tasks are determining chemical structures of compounds and rule discovery in organic chemistry. Section 3 describes two systems, MOLGEN and GENIS/HYPGENE. The first one is a system that designs experiments in molecular genetics, while the second one is a hypothesis design and theory revision system in biochemistry. Section 4 describes two well known discovery systems, AM and EURISKO, which model mathematical and formal discovery. Section 5 examines two theoretical discovery systems, PI and GALILEO. Section 6 describes several systems that model qualitative discoveries in chemistry and physics. These are GLAUBER, NGLAUBER, STAHL, STAHLp, REVOLVER, BR-3, KEKADA, AbE and COAST. In section 7, two quantitative discovery systems, BACON and IDS, are described. Finally the paper concludes with a summary of observations on these systems. There are a number of other discovery systems which are not included in this survey, but those that are included provide sufficient examples of the methods of representation, learning and discovery.

2 Molecular structure and rule discovery: DENDRAL and Meta-DENDRAL

The programs described in this section constitute some of the earliest successful examples of computational models used in scientific research. We start with DENDRAL (Buchanan & Feigenbaum, 1978), a system which determines the molecular structure of complex chemical compounds by using a set of domain heuristics. The second system, Meta-DENDRAL (Buchanan & Feigenbaum, 1978) discovers new heuristics for molecular structure elucidations.

2.1 DENDRAL

DENDRAL (Buchanan & Feigenbaum, 1978) is a theory-driven system designed to help organic chemists determine the molecular structure of unknown compounds by chemical analysis. The system incorporates large amounts of domain knowledge, and can receive experimental data. DENDRAL can determine the molecular structure of the compound by using its domain knowledge.

In a chemical analysis, an unknown chemical substance is isolated from natural sources, synthetic or pharmaceutical materials, A variety of chemical, physical and spectroscopic data are collected on the sample and structural hypotheses in the form of functional groups, or more

complex structures are generated by the interpretation of these data. These fragments are assembled into complete structures for a set of candidate structures. These candidates are examined and experiments are designed to classify them. The experiments result in new structural information which serves to reduce the set of candidate structures. Eventually, the correct structure is found among the candidates.

2.1.1 Knowledge representation

DENDRAL's descriptive domain knowledge is represented in INTERLISP in list structures such as attribute-value pairs and lists. The system's prescriptive knowledge (i.e., its domain heuristics) is represented as production rules. DENDRAL's domain knowledge includes descriptions about various molecular structures, mass spectrometric data, and rules about molecular structure elucidations from experimental data.

2.1.2 DENDRALS's operators and their interactions

DENTRAL is organized in three main parts which are characterized by a plan-generate-test sequence. Accordingly, it has three subsystems. The number of chemical structures for relevant molecular formulae can be extremely large, thus it is essential to constrain structure generation to only plausible molecular structures. DENDRAL's first subsystem examines the data to infer constraints (it uses a large amount of knowledge of mass spectrometry for this task). These constraints are given to the second module, which in turn generates all possible molecular structures satisfying the constraints. Each of the structures is tested by the third module. These operators apply a set of breaking rules to predict which bonds in the proposed structure will be broken. The simulated spectra are compared with the actual spectrum, and the structure corresponding to the best matching spectrum is ranked as the most likely structure for the unknown sample. The third module uses simple theoretical knowledge and the more subtle rules of mass spectrometry to classify the remaining structures according to the number of predicted events found (and not found) in the observed data.

The system has interfaces for research chemists to make the configuration generation easy. The interface contains a structure editor and a graphic program. DENDRAL's planning is limited to mass spectrometry, but many of its techniques are general. The system has been used to aid in structure determination problems of substances such as terpenoids, organic acids, antibiotics and hormones, DENDRAL was also able to check the accuracy of published structure elucidations of some organic compounds. In some cases, the system found structures which were plausible alternatives to the published structures.[2] A variant of DENDRAL is GA1 (Stefik, 1978), which infers DNA structures from segmentation data utilizing symmetry constraints with a simpler set of domain heuristics.

Despite its successes, from the standpoint of modelling scientific research, DENDRAL is more like an expert system operating in a narrow domain. The system is confined to problem solving, and does not have theory formation and theory revision capabilities.

2.2 Meta-DENDRAL

As the DENDRAL system determines the chemical structure of a substance by using its rules under elaborate control functions, Meta-DENDRAL (Buchanan & Feigenbaum, 1978; see also Dietterich & Michalski, 1983) has been built to discover such rules. It is an interactive program that can help chemists determine the dependence of mass spectrometric fragmentation on substructural features.

Meta-DENTRAL uses a weaker body of domain knowledge to formulate the rules of mass spectrometry by using induction over empirical data. It receives as input the structures and the

[2]See Buchanan and Feigenbaum (1978).

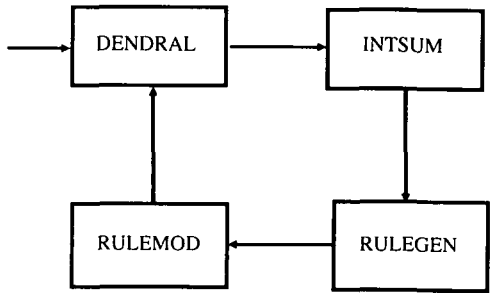

Figure 1 Interactions of DENDRAL and Meta-DENDRAL. The latter has three components: INTSUM interprets spectral data; RULEGEN generates rules; and RULEMOD evaluates and modifies these rules. (In figures throughout this article, rectangles represent system operators or operations, while round figures represent descriptive knowledge states.)

mass spectra of a set of known molecules. It decides which events (data points) are important, and looks for molecular fragmentations what will explain them. It attempts to form general rules by correlating plausible fragmentations with substructural features of the molecules and then tests and modifies them. In this way, the rules of mass spectrometry are expressed as condition-action rules where the left hand side is a description of the chemical structure of some relevant piece of the molecule and the right hand side is a list of processes which occur (or may cause these fragmentations). In summary, a rule discovered by Meta-DENDRAL would look like

molecular structure → *mass spectrum*

Descriptive domain knowledge is represented in Meta-DENDRAL, as in the DENDRAL system, in the form of list structures such as attribute-value pairs and lists. Similarly, its prescriptive knowledge is represented as production rules.

Rules are formed by Meta-DENDRAL in a three-stage sequence by its three main subsystems. The first module interprets spectral data of known compounds in terms of possible fragmentations and atom migrations. For each molecule in a given set, this module first produces the plausible processes which might occur. Then it examines the spectra of the molecules looking for evidence (spectral peaks) for each process.

After the data have been interpreted, control passes to the second module, which is a heuristic search program for rule generation. This module generates plausible rules within syntactic, theoretical and empirical constraints. It creates general rules by selecting the important features of the molecular structure around the site of the fragmentations proposed by the first module. These features are combined to form a substructure. Each substructure considered becomes the left hand side of a candidate rule whose right hand side is the proposed process. The third module evaluates the plausible rules generated by the second module and modifies them by making them more general or more specific. This program has five tasks: selecting a subset of important rules, merging rules, deleting negative cases by making rules more specific, generalizing rules, and selecting the final set. Meta-DENDRAL's operators and their interactions are shown in Figure 1.

Meta-DENDRAL has successfully rediscovered known, published rules of mass spectrometry for aliphatic compounds and steroids, and has also discovered new rules for three closely related families of structures for which rules had not previously been reported (see Buchanan et al., 1976). Later, Meta-DENDRAL was adapted to another spectroscopic technique, nuclear magnetic resonance (NMR) spectroscopy.

Meta-DENDRAL constitutes a clear advance from the DENDRAL system, as it has the capability of hypothesis generation which is essential for theory formation. However, the system lacks the theory revision capabilities as it cannot modify its hypotheses. Additionally, like the DENDRAL system, Meta-DENDRAL does not represent physical or chemical processes, but their outcomes.

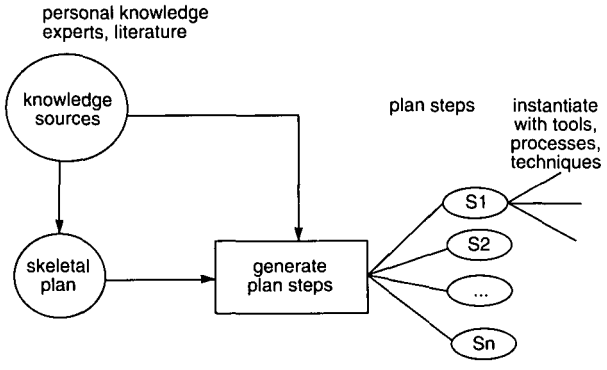

Figure 2 MOLGEN's generation and instantiation of skeletal plans and plan steps with processes, tools and techniques.

3 Planning and discovery in molecular biology

In this section two systems are examined, MOLGEN and GENSIM/HYPGENE. The first system plans and designs experiments in molecular genetics by using its knowledge of experiment design rules. The second program is a composite system that formulates hypotheses about chemical processes or sequences of such processes by utilizing planning techniques.

3.1 MOLGEN

Genetics researchers carry out large numbers of experiments for structural analysis to determine the physical and chemical structure of DNA molecules. The experiments are designed by using basic theoretical knowledge of molecular biology, knowledge of known structures, knowledge about laboratory methods and basic analytic strategies.

MOLGEN (Friedland, 1979; Stefik, 1981 a;b) is a theory driven system which designs experiments in molecular genetics by using its knowledge base for known strategies and techniques to achieve a goal determined by the user. The program simulates reseachers in achieving a research goal by choosing an appropriate strategy using external knowledge (i.e., relevant literature, domain experts) and internal knowledge. The strategy consists of a skeletal plan generalized from past experience. MOLGEN then instantiates each of the steps of its plan with a specific process, laboratory tool or technique. It uses three types of criteria in selecting the process, tool or technique: it must be possible to accomplish the goal with it; it must work under particular laboratory conditions; and it must be more convenient, more reliable, more accurate, less time consuming and less costly than competing choices.

MOLGEN's domain knowledge is represented in frames. For example, DNA structure frames contain slots for information relevant for DNA structures. They also contain rules for checking the consistency of information and for filling in additional slots by inheritance. The skeletal plans are represented in frames that contain the description of the plan and its possible general and specific uses. Bulk of the domain knowledge consists of the descriptions of the laboratory techniques. Techniques are described in a unit containing information about the utility, optimal laboratory conditions and the selection rules relevant to the technique.

MOLGEN generates and instantiates skeletal plans and plan steps as illustrated in Figure 2. Each plan step is matched with choices (processes, tools, techniques). If the match fails, the plan step is reformulated or subgoals are generated. MOLGEN uses five general strategies in experimental research: the process, tool or technique must be relevant to the goal; it must be applicable; if more than one option is available, then choose the best option against reliability, accuracy, time and cost; reformulate the goal or strategy when there is no process, tool or technique found to accomplish it; or create subgoals or substrategies.

MOLGEN is not really a discovery system, but is concerned with hierarchical planning of gene

cloning experiments under constraints. Like the DENDRAL system, MOLGEN performs only one of a series of activities that take place in scientific research. The system relies on its expert knowledge in planning, and lacks the capabilities of revising its domain knowledge. However, it highlights the importance of experimental planning in a complex field of empirical research. Some of its methods and strategies have been implemented later in KEKADA (Kulkarni & Simon, 1988).

3.2 GENSIM/HYPGENE

GENSIM/HYPGENE (Karp, 1990), is a composite system that constructs hypotheses about chemical processes in a specialized area of biochemistry. The system treats hypothesis generation as a design problem, and uses a planner in such hypothesis designs. Unlike the earlier programs, GENSIM/HYPGENE has theory revision capabilities, albeit operating in a narrow domain, in the discovery of bacterial gene regulation.

3.2.1 Knowledge representation

The first component of this composite system, GENSIM, represents theories and experiments as process schemas, and domain objects in a taxonomic hierarchy of frames. The system maintains its domain knowledge in three different knowledge bases: 'class knowledge base', 'simulation knowledge base', and 'process knowledge base'. The first one represents GENSIM's formal knowledge (i.e., taxonomic biological knowledge) and object descriptions. The second one contains the knowledge of chemical reactions between the biological entities and the descriptions of the experiments. Finally, the last one includes the system's knowledge about qualitative processes of its domain using a representation similar to Forbus's (1984) qualitative process schemas. Figure 3 summarizes GENSIM's description of a domain process. GENSIM predicts the outcomes of experiments using a rule-based interpreter that applies processes to the objects (e.g., chemical compounds) present in the experiments.

The other component, HYPGENE, formulates hypotheses by using GENSIM's knowledge base by a set of design operators. The inputs of HYPGENE are arbitrary predicate statements and definitions. These include a description of the initial conditions of an experiment, GENSIM's prediction of the outcome of the experiment, a description of the error in this prediction, and the domain theory used in the prediction. The subsystem's output is a set of hypotheses which are used in revising GENSIM's theory. Figure 4 summarizes the interactions of GENSIM/HYPGENE's components.

HYPGENE's design goals specify three types of problem states between GENSIM's predictions and the experiment results: objects with certain properties should be added to or removed from the prediction; properties of existing objects in the prediction should be modified; the concentration of objects in the prediction should be modified. An example hypothesis generation task for HYPGENE is as follows:

The observed rate of a particular substitution reaction is higher than that predicted in the current experiment. Generate hypotheses to explain this observation.

This goal asks the designer to modify the initial conditions of the process, or the domain theory.

Parameters:	*A*, *B*
Preconditions:	*B* contains an active site *S*,
	S interacts with *A*,
	S is not occupied,
	S does not contain a mutation that disables the current reaction.
Effects:	A new object *C* is created with certain properties.

Figure 3 GENSIM's process description of a reaction between a certain protein and a region of DNA. (The process description is given in general terms to illustrate knowledge representation in a process schema.)

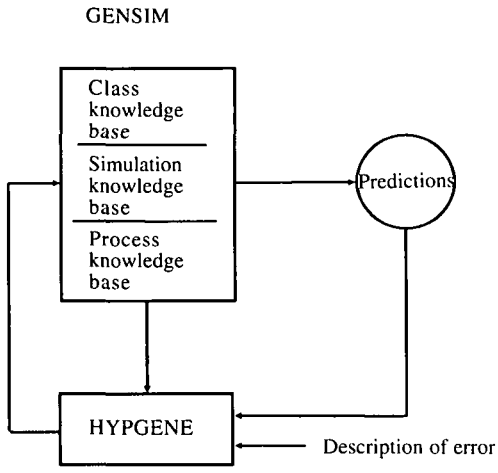

Figure 4 Interactions of GENSIM/HYPGENE's components. HYPGENE compares GENSIM's predictions with experiment results and generates new hypotheses for the latter.

3.2.2 Methods of theory revision and discovery

When GENSIM's predictions contradict the outcomes of an experiment, theory revision takes effect. The system uses backward chaining from the experiment results to the process conditions. In revising hypotheses, HYPGENE uses a combination of heuristic, best-first search and assumption-based belief revision (de Kleer, 1986) methods. The system's theory revision basically consists of making changes in the process descriptions in accordance with the possible differences between predictions and experiment results.

To illustrate, consider a process which involves a series of reactions such as $A + B \rightarrow C$, $C + D \rightarrow E$. In this process, when A, B, C and D are present and the reaction conditions are satisfied, E is obtained. Suppose that, contrary to expectations from the system's process knowledge base, E is not obtained in the process despite the presence of A, B, C and D. GENSIM/HYPGENE has to explain this phenomenon. The system considers the sub-process $C + D \rightarrow E$, and tries to find out if any of the process conditions of this reaction is violated. If one or more conditions are violated, then this would explain the absence of E in the medium. If no condition is violated, then the program makes changes in the process conditions so that the absence of E is explained. Since there can be several process conditions, a number of alternative process modifications are possible. In order to reduce the search, these modifications are guided by the system's domain knowledge. GENSIM/HYPGENE improves its domain theory by revising its process descriptions in this way.

GENSIM/HYPGENE is essentially a knowledge intensive theory formation system. The prominent feature of this program is its inclusion of a planner, which enables it to reason about a series of interlinked processes with complex propositions. Theory formation and revision is only part of a series of activities that lead to scientific discovery. Therefore, GENSIM/HYPGENE has a limited scope as an empirical discovery system. The program cannot evaluate its input data (i.e., the experiment results) and accepts them as correct. However, the system's theory revision methods are applicable to other domains in physical science as they rely on making syntactic changes in the qualitative process descriptions.

4 Mathematical and formal discovery: AM and EURISKO

In this section two systems are examined. The first program is AM (Lenat, 1979), which discovers a number of mathematical concepts from a set of elementary set theoretical concepts and a series of condition-action rules or 'heuristics'. The second system, EURISKO (Lenat, 1983 a), is an improvement on the former, in that it not only can discover new mathematical and formal concepts, but also new rules that are used in such discoveries.

4.1 The AM system

AM (Lenat, 1979) is a theory driven discovery system which investigates and discovers concepts in elementary set theory and number theory. Starting with a series of condition-action rules (heuristics), and a series of concepts selected from finite set theory, the system is capable of creating, defining and evaluating new concepts on the basis of their relevance by using its rules over the domain concepts via an agenda mechanism. AM searches for regularities in the data about its concepts (i.e., their instances) and hypothesizes conjectures relating them to one another.

The reasons for choosing AM's domain as set theory and arithmetics given by Lenat (1979) is as follows: there are no uncertainties in the data given to the system; choosing a familiar field eliminates the reliance on experts' knowledge; formal knowledge is easier to automate; a mathematician is free to explore without a specific goal; elementary mathematical research has an abundance of powerful heuristic rules available; agreement on the thesis that artificial intelligence can succeed in automating only those activities for which there exists a 'strong theory' of how that activity is performed by human experts.

4.1.1 Knowledge representation

AM is a system which blends frame representation with rule-based representation. It has 242 heuristics which are represented as condition-action rules. The condition part tells us under what conditions the rule should be executed, and the action part carries out tasks such as creating a new concept or finding examples of an existing concept.

Initially, the system has 115 concepts. These are represented in frames, with each frame having 25 slots. The number of slots are fixed once and for all, but they cover enough space to represent various aspects of the domain concepts. AM's concepts can be objects (e.g., sets, bags), relations (e.g., constant, equivalent) and activities (or functions, e.g., set-union, set-intersect). A high-level description of a simplified example of AM's representation of the concept 'set' is as follows:

Name: set, proper collection, proper class
Definitions:
[A recursive definition of 'set'.]
[A simpler recursive definition of 'set'.]
[A definition of 'set' for fast search.]
Specializations: empty set, nonempty set, singleton, doubleton
Generalizations: unordered structure, collection
Examples:
Typical examples: {{}}, {A}, {A,B}, {3}
Rare examples: {}, {A,B,{C,{{{A,C,(3,3,9),,[B],A}}}}}
Conjectures: All unordered structures are sets.
Analogies: There is an analogy between sets and lists in relation to the operations applied to each. (i.e., set operations and list operations are analogous.)
Worth: 600 [on a scale of 0–1000]
Suggest: If P is an interesting predicate over X, then consider $\{x \in X | P(x)\}$
Applicable operators: union, intersection, set difference, subset, member, cartesian product, set equality
In the range of: union, intersect, set difference

Each of the 115 initial concepts is given a 'Worth' value by the 'user'. AM's rules are attached to the slots of AM's concept frames. For example, the following rule is attached to the *EXAMPLES* slot of AM's most general concept *ANY-CONCEPT*:

> *If the current task is 'Fill in examples of X', and X is a specialization of some concept Y, then apply the definition of X to each of the examples of Y, and retain those that satisfy the definition.*

Thus, in contrast to an ordinary production system where the condition part of each rule is

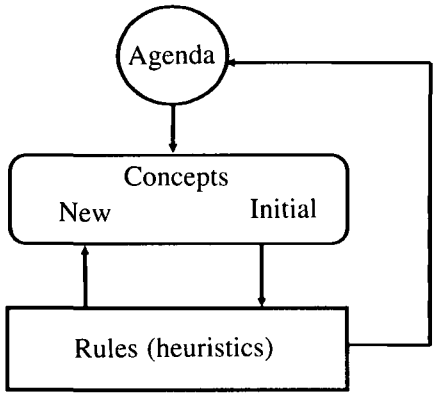

Figure 5 AM's heuristics and their interaction with its domain concepts and agenda.

compared to the contents of a working memory and all rules that match are executed, AM is much more selective about which rules it executes.

4.1.2 AM's control structure

AM runs in cycles and uses a heuristically guided best-first search. Accordingly, when it is active the top task on the agenda is considered first. The program allocates resources (time and space) for each task. AM's basic activity is to find entries for some slot of some concept. As shown in Figure 5, it operates from an agenda of tasks of the form: 'Check (or fill in) slot *S* of concept *C*'. AM's agenda is a list of slot/concept pairs. Each task on the agenda is assigned a priority rating, based on the number and strength of reasons supporting it. A typical entry on the agenda can be a task as follows:

Activity: Fill in some instances
Slot: for the *GENERALIZATIONS* slot
Concept: of the *PRIMES* concept
Reasons: –There is only one generalization of *PRIMES* so far.
–The worth rating of *PRIMES* is now very high.
Priority: –350 [on a scale of 0–1000]

AM's simple control operator chooses the task with the highest priority rating on the agenda and then executes it. During the execution of tasks, some new tasks may be proposed and added to the agenda, some new concepts may be created, and some entries for the specified slot of the current concept are found and filled in.

When a task is chosen, the agenda specifies which concept *C* and which slot *S* are to be worked on. AM first looks at the slot *S* of concept *C* to see if any rules are attached to it, then at the slot *S* of each generalization of *C*, at each of their generalization, and so on. In this way, as all the concepts of AM are arranged in a tree structure, all its rules are automatically arranged in the same way, in association with the concepts. So there is a hierarchy induced upon the set of rules and inheritance principles hold: a heuristic associated with a concept is applicable to all specializations of that concept. When the current task is related to a concept, AM gathers all the rules associated with the domain concepts which are generalizations of that concept. Figure 6 shows AM's techniques of generating extensions of its domain concepts.

A rule is activated when all of its conditions are satisfied. As shown in Figure 7, when one of AM's rules is activated, it performs one or more of the following tasks: it fills in slot *S* of some concept *C*; checks slot *S* of concept *C*; creates and defines a new concept; adds a new task to the agenda (e.g., 'Fill in examples of *PRIMES*'); modifies the interestingness of a task on the agenda. Figure 8 summarizes AM's tasks in sequence.

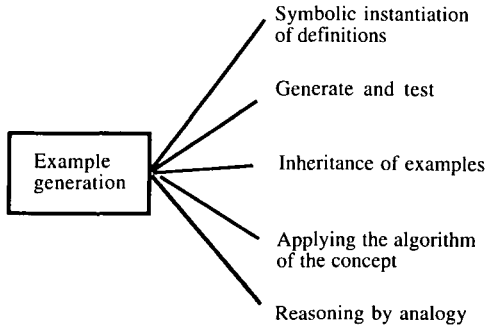

Figure 6 AM's techniques for generating the extensions of its concepts. The program generates the examples by symbolic instantiation, by its 'generate and test' rule, inheritance, by applying the algorithm of the concept, and by analogy.

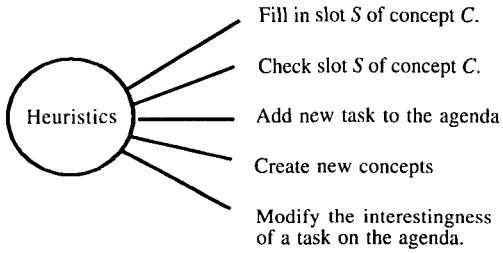

Figure 7 A heuristic rule of AM can do the following actions: Filling in slot S of some concept C, checking slot S of C, adding a new task to the agenda, creating a new concept, and modifying the interestingness of a task on the agenda.

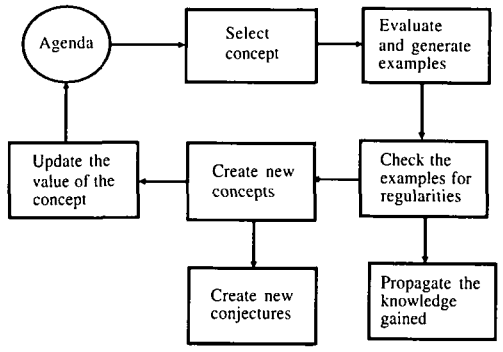

Figure 8 View of AM's activities.

4.1.3 AM's discoveries

AM can rediscover a series of mathematical concepts such as 'integer', 'prime number', 'square' and 'square root'. In a typical run of a few hours, AM defines 200 new concepts, half of which are quite well known in mathematics such as the concepts of natural number and prime number. During its operations, AM moves from finite set theory to divisibility and number theory. Besides many known arithmetical concepts, AM was able to discover some new ones such as maximally-divisible numbers. The system can also rediscover the fundamental theorem of arithmetic (i.e., every natural number has a unique prime factorization) and Goldbach's conjecture (i.e., every natural number can be expressed as the sum of two primes). Below is a summary of AM's activities when it discovers the concept of primes. AM starts with the task of filling in the examples of the concept 'Divisors-of':

Current Task: Fill in examples of the concept 'Divisors-of'.
3 Reasons:

(1) No examples of Divisors-of are known yet.
(2) The concept 'Times' is now very interesting.
(3) Focus of attention: AM just defined Divisors-of.
26 examples found, in 9 seconds. E.g., Divisors-of(6) = {1,2,3,6}

Current Task: Consider numbers having small sets of Divisors-of.
2 Reasons:

(1) It it worthwhile to look for extreme cases.
(2) Focus of attention: AM just worked on Divisors-of.

Filling in examples of numbers with 0 divisors.
  0 examples found in 4.0 seconds.
  Conjecture: No numbers have 0 divisors.
Filling in examples of numbers with 1 divisors.
  1 example found in 4.0 seconds. e.g., Divisors-of(1) = {1}.
  Conjecture: 1 is the only number with exactly 1 divisor.
Filling in examples of numbers with 2 divisors.
  24 examples found in 4.0 seconds. e.g., Divisors-of(13) = {1,13}.
  No obvious conjecture. May merit more study.
  Creating a new concept: 'Numbers-with-2-divisors'.
Filling in examples of numbers with 3 divisors.
  11 examples found in 4 seconds. e.g., Divisors-of(49) = {1,7,49}.
  All numbers with 3 divisors are also Perfect-squares.
  Creating a new concept: Numbers-with-3-divisors'.

After this, AM continues with the tasks of considering the square roots of numbers with 3 divisors and squares of numbers with 3 divisors.

The AM program demonstrates that a relatively small number of general rules can guide a nontrivial discovery process, at least in formal sciences. The limits of the program can be summarized as follows: AM's main goal is to find interesting new concepts from a set of formal concepts. It's search is, therefore, limited by the set of tasks it can perform. Since it is a formal discovery system, it does not represent most of the processes involved in empirical discovery such as experimental design, conducting experiments, data collection and evaluation, empirical hypothesis formation, and theory revision.

AM does not synthesize new rules for dealing effectively with the concepts it generates. It has no way of improving its rules. Lenat (1983 a; b) proposed a solution to this problem with his later project EURISKO. The EURISKO system treats its rules as concepts, and uses meta-rules to reason about its rules and generate new rules from them.

4.2 EURISKO

Theory formation constitutes an important part of scientific activity. Lenat (1983 a) describes the tasks in theory formation for discovery systems as follows:

1 When some new definitions, objects, operations, rules, etc., are given, gather relevant empirical data.
2 Detect regularities, patterns and exceptions to these in the data.
3 form new hypotheses and modify old ones in accordance with the observations. Design and carry out experiments to test these hypotheses.

4 As the number of hypotheses increases, combine them if possible, by making new definitions. Repeat steps 1–4.

These strategies and rules had been used by the AM system albeit in a limited, formal sense. Lenat's (1983 a) EURISKO system has the following additional capabilities:

5 As the above processes (1–4) proceed, generate some new rules (heuristics) by compiling the learner's experience.
6 Augment or change the representation of the domain knowledge when necessary.

AM's 'objects' were formal concepts (e.g., set theoretical and arithmetical concepts). Like AM, most of the fields which EURISKO explored are formal or formalizable domains of knowledge. The regularities mentioned in Step 2 can be qualitative and/or quantitative.

Step 2 above assumes that rules can be synthesized or modified just as other operations of the system. Step 6 assumes that the representation of knowledge can be changed whenever necessary by using a set of rules. EURISKO can define a new slot for its frame-like language. A set of rules and metarules can guide the discovery, evaluation and modification of rules. An example (see Lenat, 1983 a, p. 54) of such a metarule is as follows:

If slots s1 and s2 of frame F can have the same type,
then define a new rule, attached to F, that says:
'If f is an interesting F and its s1 and s2 are of the same type,
then define and study the situation in which f's s1 and s2 values are equal.'

This metarule leads to many useful rules to be synthesized. It has been applied to functions and relations yielding interesting concepts such as reflexive and anti-reflexive relations and doubling, squaring, etc., functions. EURISKO uses a new language for its rules instead of LISP codes which AM has to rely on for the successful operation of its rules. Its vocabulary includes many types of conditions, actions and descriptive properties that a rule might possess.

Lenat (1983 b) describes the design, control and communication ideas which led to the construction of the EURISKO system. For reasons of space, only his control ideas will be described here. Lenat argues that the control structure of the system should be represented as part of the knowledge base. Explicit representation of the control structure has three benefits: it facilitates explanations, it allows to enforce constraints to the activities of the heuristics and it allows the system to detect and avoid infinite loops. Lenat has abandoned the original idea of enabling the program to meaningfully modify its own control code, for it had resulted in bugs. Elsewhere (Kocabas, 1991 b) I have discussed how such an idea can be implemented in a hierarchic homuncular system by using different learning methods without the undesired effects.

Lenat also proposes to maintain multiple agenda in complex discovery systems. EURISKO has several 'topics' to investigate (e.g., games, device physics, number theory). Each topic has an 'Agenda' slot which contains its own agenda of tasks dealing with that topic and its specializations. Also, the system should be capable of creating and eliminating its agendas dynamically. It should be able to select and execute a task. It should be able to carry out task analyses after the execution of tasks as necessary. Lenat also suggests that a system's heuristics should be regarded as concepts, and there should be no distinction between rules and metarules.

EURISKO constitutes a clear advance from its predecessor, as it tackles one of the most difficult tasks in discovery, namely the discovery of general rules. The system has been applied to Naval Fleet Design, elementary mathematics, LISP programming, evolution, games, VLSI design, heuretics (the study of heuristics) and representation. In the nationwide Naval Fleet Design games, Lenat with EURISKO won the 1981 and 1982 tournaments in the USA. In its application to evolution, the EURISKO experiments suggest that the simulated evolution hardly progresses when mutation was random, but quite rapidly when mutation is under the control of a set of heuristics. However, despite its successes, EURISKO remains a limited model of scientific discovery, as it lacks many of the elements of empirical discovery.

5 Theoretical discovery: PI and GALILEO

Theoretical discoveries result from theoretical analysis and synthesis of domain knowledge. Various kinds of reasoning such as deduction, abstraction, abduction, conceptual combination and analogy is applied in theoretical analysis. In this section two systems are described: the first is PI (Thagard & Holyoak, 1985), which simulates the discovery of the wave theory of sound by carrying out a form of conceptual analysis. The other system, GALILEO (Zytkow, 1990), transforms quantitative scientific laws into more generally applicable forms by algebraic methods.

5.1 The PI system

PI is a theory driven system which models the discovery of the wave theory of sound. Like the AM system (Lenat, 1979) its research activity is confined to the area of conceptual study, but its techniques are different from that system. Although the scope of PI is limited, it addresses an important task in the processes of scientific discovery: theoretical analysis. One of the aims of theoretical analysis in science is to combine empirical results into a coherent body of theoretical system. This usually helps to express scientific knowledge in a small set of formulae. History of science has many examples of natural phenomena previously regarded as distinct and treated separately in science, subsequently explained as different manifestations of a common principle or law (one such example is electricity and magnetism).

5.1.1 Knowledge representation

PI's initial knowledge includes information about concepts such as 'sound', 'reflects', 'wave', 'propagate' and 'motion' (see Thagard, 1988). Knowledge is represented in PI in frames and rules attached to them, much like in the AM system. Domain concepts are represented as attribute value pairs in frames. The rules of PI can have any number of conditions and actions, which are represented in predicate calculus notation. Thagard & Holyoak (1985) do not make a clear distinction between inference rules and action-rules in their presentation, in the sense that they say 'rules' when they mean 'descriptive hypotheses' generated by the PI system.

5.2.1 PI's control structure

PI's rules serve to explain a phenomenon by analogy, instance-based generalization, condition-based generalization, and a combination of concepts and abduction. In the instance-based generalization, PI infers that all A are B on the basis of instances of A that are also instances of B. In PI such inductions are triggered when the dynamic memory of the system contains the information that two or more objects are instances of the same two active concepts.

Condition-based generalization is performed by taking the intersection of the conditions of two inference rules. For example, from the conditionals

$$A(x).P(x) \rightarrow Q(x),$$
$$A(x).R(x) \rightarrow Q(x),$$

PI infers $A(x) \rightarrow Q(x)$.

Abduction involves the generalization of hypotheses to provide potential explanations of puzzling phenomena. In PI, finding an explanation to a phenomenon is considered as a system goal. Abduction can be stated in predicate logic symbolism as follows:

$$P(x) \rightarrow Q(x),$$
$$\underline{Q(a)},$$
$$P(a)$$

which can be stated in different terms: if $P(x)$ causes $Q(x)$ and $Q(a)$ is observed, then $P(a)$ is the cause.

Conceptual combination is performed by combining existing concepts, by comparing the 'topics' of the concepts for conflicting values. If no conflicting values are found, then the combined concept would not be an interesting one, and hence no further processing is required. If a conflict is found, this triggers the formation of a new concept to represent the combination.

5.1.3 Simulation of the discovery of the wave theory of sound

PI simulates the discovery of the wave theory of sound, which was first discussed by Vitruvius (see Thagard & Holyoak, 1985) around the first century A.D. The discovery is thought to have occurred in the context of trying to explain why sound propagates and reflects and noticing an analogy with water waves.

PI proceeds like the AM system which starts with activating its concepts. The system starts with the activation of the concepts 'sound' and 'reflects'. Activation of the latter leads to the activation of several hypotheses concerning reflection, including the information that water waves reflect and rope waves reflect. By condition-based generalization, the hypothesis that all waves reflect is generated. By abduction from 'if x is a wave, then x reflects' and 'sound reflects', the conclusion 'sound is a wave' is arrived at. This triggers both the generalization that all sounds are waves and the conceptual combination of sound and wave. The result is the new theoretical concept 'sound-wave' which contains the information that sound is a kind of wave as well as numerous new hypotheses. Having formed the hypothesis that sound consists of waves, PI is able to deduce why the sound x reflects, so the problem of explaining why sound reflects is solved.

The PI system is concerned with only one of the tasks involved in scientific research: theory formation by conceptual analysis and synthesis. Therefore, its scope is very limited in terms of scientific discovery. Its search space does not seem to be very large. Despite these limitations, and the problems PI is concerned with may seem trivial, it is nevertheless an attempt to illustrate how conceptual analysis and synthesis is carried out to perform theoretical generalizations, and to produce new and theoretically useful concepts. Recently, Thagard and Nowak (1990) describe a much more sophisticated system, ECHO, which carries out a detailed conceptual and theoretical analysis of the scientific developments that led to the modern geological theory in this century.

5.2 GALILEO

Zytkow's (1990) GALILEO system addresses an important problem in theory development. Scientific laws are meaningful in reference to the structure of physical situations, whereas BACON-like systems discover laws applicable to only a limited set of physical situations. The difference is more apparent in complex physical systems. To illustrate, consider a simple electrical circuit which is described by the equation $I = E/(R + r)$ where I is the electric current, E is the electromotive power of the battery with an internal resistance r, which supplies power to a resistor R. This equation can be rewritten as $E = IR + Ir$, which is more expressive from the standpoint of qualitative process analysis. If more elements are added to the circuit, the equation becomes

$$E + E_1 + \cdots E_n = IR + Ir + Ir_1 + \cdots Ir_n.$$

This example shows how scientists deal with the complexity of physical situations by transforming and combining laws that represent simple situations. GALILEO carries out such theoretical analysis and synthesis by using the qualitative representation of the physical situations, and the relationships between these and the terms of the scientific law.

5.2.1 Knowledge representation

GALILEO mainly has three types of knowledge: knowledge about qualitative processes, quantitative laws, and a series of algebraic and heuristic operations. Qualitative processes are represented in 'process diagrams' based on Forbus's (1984) and Nordhausen and Langley's (1987) representation of qualitative processes.

Quantitative laws are represented as symbolic equations in the form of lists, so that various

algebraic transformations and matching operations can easily be applied to them. Algebraic and heuristic operations are represented by condition-action rules and procedures. GALILEO represents its process diagrams by a process decomposition tree, and the quantitative laws by equation parse trees. The system receives as input the expression of a quantitative law and the qualitative process diagrams. The user also specifies what simple terms represent measurements in each state individually.

5.2.2 GALILEO's control structure

GALILEO tries to find the equation form that matches the process decomposition tree generated from the process diagram. It uses a depth first search method with backtracking, combined with heuristic search. In general, the search space is not large, so the adopted search methods are sufficient.

The system uses the process decomposition trees to guide its transformations of the equations that represent quantitative laws. One process diagram may be represented by several process decomposition trees. GALILEO's matching procedure searches for a mapping between the branches in the process decomposition tree and the branches in the equation parse tree. Each term in the equation is then attached to the matching element in the process diagram.

The program uses a variety of transformation rules to modify the parse tree. Most of these are simple algebraic rules such as $(a + b)/c \rightarrow a/c + b/c$. Transformation rules are activated by a matching between their condition parts and the current equation or one of its parts. Equation transformation continues until a complete match with the process decomposition tree is established, or no improvement can be made to a partial match. The first case indicates that the search goal has been achieved, while if the match is not complete, the search returns a partial match, which aids in the selection of operators for the next step in the transformation search.

GALILEO's operations are limited to theoretical analysis and synthesis, but its methods are general, and are expected to be incorporated in more comprehensive models of discovery. Theoretical analysis is a neglected area in machine discovery despite its importance. For example, it would be interesting to have a system capable of modelling Maxwell's theoretical work in electromagnetism. Similarly, another problem to be tackled is the formulation of the special theory of relativity through Einstein-Lorenz transformations.

6 Qualitative discovery and theory revision in physics and chemistry

In this section, several systems are examined which model various qualitative discoveries and theory revision in physics and chemistry. These systems are GLAUBER (Langley et al., 1987), NGALUBER (Jones, 1986), STAHL (Zytkow & Simon, 1986), STAHLp (Rose & Langley, 1986), REVOLVER (Rose & Langley, 1988), BR-3 (Kocabas, 1991 a), KEKADA (Kulkarni & Simon, 1988), AbE (O'Rorke et al., 1990) and COAST (Rajamoney, 1990). With the exception of the first two, these systems have the capability of revising their domain theory when contradictions arise with observations.

6.1 GLAUBER

GLAUBER (Langley et al., 1987) is a data driven system that models the discoveries of simple qualitative principles in the 17th century. In the 17th and 18th centuries, chemists were able to classify substances according to their qualitative properties. They focused on such features as physical state, colour and taste, as well as their interactions with other substances. From such properties, they defined classes such as acids, alkalis and salts. Correlating these features with their reactions, chemists began to formulate the first qualitative principles of chemistry.

A 17th century German chemist, J. R. Glauber, was one of the chemists who played an important part in the development of the acid–base theory. The GLAUBER program models the discovery of the principles of this theory. I have written a somewhat simpler Prolog version of this

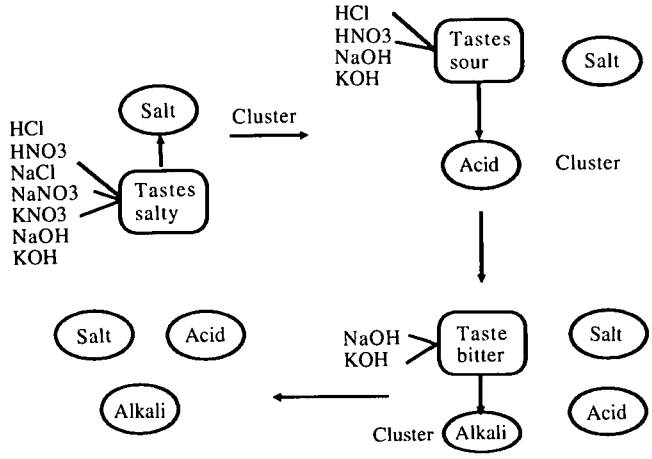

Figure 9 Conceptual clustering of GLAUBER, in which the chemical substances are grouped by their distinguishing properties.

program, named P-GLAUBER, and will use this program in explaining GLAUBER's representation and operations, stating the differences between the two systems where necessary.

6.1.1 Knowledge representation

Knowledge is represented in GLAUBER in predicate logic expressions. Each domain fact or observation is expressed by a predicate followed by labelled arguments. P-GLAUBER's representation in the object level is slightly simpler. In P-GLAUBER the fact that HCl reacts with NaOH to form NaCl is represented by a predicate expression whose arguments constitute an ordered tuple of sets of the reactant and resultant substances

reacts([HCl,NaOH],[NaCl]).

Statements which express the properties of substances, such as 'Hydrochloric acid tastes sour', are represented as:

tastes_sour(HCl).

From a series of statements about the reactions of several substances and propositions about their properties, P-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 classes of substances to provide generality. For example, the qualitative law

reacts([acid,alkali],[salt]).

has the same form as

reacts([HCl,NaOH],[NaCl]),

but HCl has been replaced by 'acid', NaOH by 'alkali' and NaCl by 'salt'.

6.1.2 GLAUBER's rules and methods of discovery

P-GLAUBER has three operators, one for forming classes of substances, one for clustering the substances according to the classes, and another one for finding hypotheses. These operators are called CLASSIFY-PROPERTIES, CLASSIFY-SUBSTANCES and FIND-HYPOTHESES, respectively.

When data about chemical substances concerning their properties and their reactions are given, P-GLAUBER first forms the classes of substances based on their properties as depicted in Figure 9. Then the program forms the clusters of its domain objects (the chemical substances) by identifying each substance with its cluster name as, for example, 'acid(HCl)', 'alkali(NaOH)'. The cluster

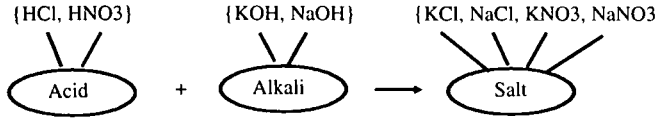

Figure 10 GLAUBER's rediscovery of the acid–base reaction principle.

names 'acid', 'alkali', etc., are either given by the user or by some symbolic name that is generated automatically by the program.

When the conceptual clustering operation is complete, P-GLAUBER searches for general qualitative principles about the reactions of the substances. It uses the representational forms of chemical reactions in its knowledge base for finding these principles by generalizations, as shown in Figure 10. Its FIND-HYPOTHESES operator checks the knowledge base for each reaction to satisfy the general condition

If for every reaction $X + Y \rightarrow Z$ where X, Y, Z are substances and X and Y belong to the clusters C1, C2 respectively, there is a cluster C3 such that Z belongs to C3, then generalize reaction $X + Y \rightarrow Z$ into $C1 + C2 \rightarrow C3$.

P-GLAUBER generates qualitative laws such as 'acids and alkalis react to give salts,' 'organic acids and alcohols react to give esters,' 'organic acids and alkalis react to give organic salts', and 'acids and metals react to give salts'.

Let us now return to the differences between P-GLAUBER and the original GLAUBER. First of all, P-GLAUBER uses a knowledge organization based on the knowledge organization principles based on Kocabas (1989). Second, the original GLAUBER system has a slightly more detailed object level representation of domain facts and relationships. The facts 'reacts([HCl, NaOH],[NaCl])' and 'tastes_sour(HCl)' are represented in GLAUBER, respectively, as

(reacts inputs {HCl NaOH} outputs {NaCl}),
(has-quality object {HCl} tastes sour).

Third, GLAUBER discovers the qualitative laws in stages. From the data on chemical substances and their reactions, the program first hypothesizes, e.g., that acids react with substances like NaOH and KOH to form salts, and then arrives at the principle that acids react with alkalis to form salts. To avoid overgeneralizations, GLAUBER uses universal and existential quantifiers in its formulation of laws and formulates the acid-base law as follows: 'For all alkalis and for all acids, there exists a salt such that when they react they form the salt'. GLAUBER employs one operator for defining classes and a second operator for proposing quantifiers. These operators are called FORM-CLASS and DETERMINE-QUANTIFIER.

GLAUBER is a data driven discovery system dependent for its discoveries on the data given to it. It assumes that all data are present at the outset. It is unable to respond to new data, even if these disconfirm the hypotheses it has formed. In contrast, P-GLAUBER can evaluate new data and can add new hypotheses by using its rules of discovery.

NGLAUBER (Jones, 1986) is an improved version of GLAUBER with an ability of proposing experiments by predicting future data. Its representation of facts and relationships is similar to GLAUBER's, with the additional capability of quantifying its assertions about its domain objects such as: 'All substances which belong to the class of salts, taste salty'. NGLAUBER can identify four types of entities: facts, nonfacts, predictions, classes. 'Facts' and 'nonfacts' are statements which it knows as true and false, respectively. A 'prediction' is represented as a pair of statements, the first of which is a proposition and the second is a hypothesis which makes the proposition true when it is true. An example is

prediction: KCl tastes salty,
All substances which belong to the class of salt, taste salty.

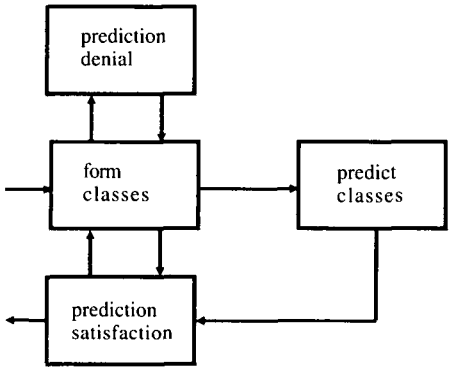

Figure 11 NGLAUBER's four main tasks concerning data: introduction, prediction, prediction satisfaction and denial.

Figure 11 shows NGLAUBER's four main tasks and how they are related with one another. Unlike its predecessor, NGLAUBER is an incremental discovery system. It does not have to be given all data at once to be able to make its discoveries. The system has four tasks to perform on data entry; introduction, prediction, prediction satisfaction and denial. The introduction mechanism forms classes, while the prediction mechanism makes predictions about the domain objects. The prediction satisfaction mechanism checks new data if new facts were predicted by the system. When a predicted fact is introduced, the predictions of that fact are removed.

The prediction denial mechanism on the other hand, checks if any fact contradicts a prediction. In case of contradictions, it deletes its predictions or reduces their generality. However, such conflicts between factual statements and hypotheses that occur in knowledge systems could be resolved by simply making a categorical distinction between hypothetical and factual statements, and by giving priority to the latter over the former (see Kocabas, 1989). NGLAUBER's discovery of the laws on chemical reaction is similar to that of GLAUBER.

In summary, the main improvement of NGLAUBER over its predecessor is its ability to make predictions. The program proposes experiments for its predictions to be tested externally, and accepts the results. NGLAUBER was one of the earliest systems to have this capability. The GLAUBER series of programs is limited to representing simple examples of theory formation, with little or no capability of theory revision. The following sections describe some of the systems with this capability.

6.2 STAHL

The STAHL system (Zytkow & Simon, 1986) is a data driven model with theory driven features of discovery. It has been developed to simulate the construction of componential models in the 18th century by chemists. One of the major goals of the 18th century chemistry was to determine the components of chemical substances. STAHL generates componential models from chemical reactions. As an improvement to the previous discovery systems, the program has a theory revision capability.

STAHL's main concern is the formulation of componential models. A typical example of a componential model is 'marine acid consists of inflammable air and chlorine'. (In today's terminology this means 'hydrochloric acid consists of hydrogen and chlorine'. However simple such componential models were, it was very difficult to construct a coherent set of such models for all the substances known at the time. There were two different theories to account for combustion processes. The earlier one was the phlogiston theory developed by G.E. Stahl, and the other one was the oxygen theory of A. Lavoisier, which several decades later replaced the former.

The phlogiston theory was based on the ancient view that fire, heat and light are different manifestations of a common principle that leaves a substance during combustion. Stahl called this principle 'phlogiston', and initiated an extensive use of the concept in reasoning about chemical reactions. In the phlogiston theory, any reaction involving combustion was viewed as a chemical

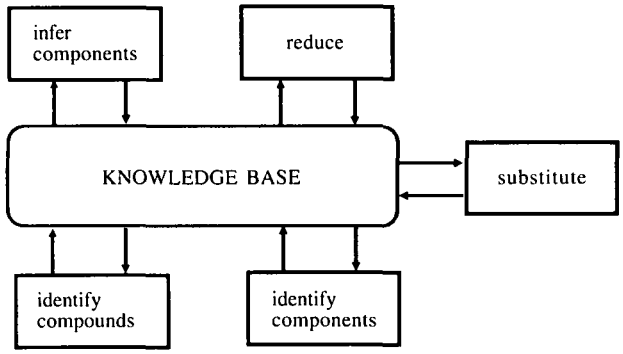

Figure 12 STAHL's rules and their interactions. The figure also suggests that, in principle, STAHL's rules can be active in parallel.

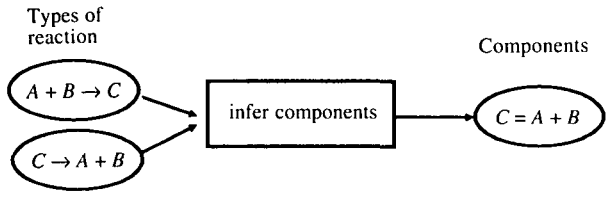

Figure 13 STAHL's *infer-components* rule in operation. The rule infers components from reactions of the form $A + B \rightarrow C$ and from decompositions like $C \rightarrow A + B$.

decomposition. For example, Stahl interpreted the burning of charcoal as its decomposition into the matter of fire (phlogiston) and ash.

Having given a summary of the background of the ideas which led to discoveries in 17th century chemistry, we can now look at the methods of the STAHL system. I will dwell on the knowledge representation and discovery methods of this program in some detail because most of its methods have since been adopted by several other systems as described below.

6.2.1 Knowledge representation

STAHL's domain knowledge includes a set of chemical reactions known to 18th century chemists. The system uses the same representation as GLAUBER's for chemical reactions. Some examples of these reactions are

(reacts inputs {charcoal air} outputs {phlogiston ash air}),
(reacts unputs {vitriolic-acid potash} outputs {vitriolated-tartar}).

STAHL is an incremental discovery system. Its inputs are qualitative facts (i.e., chemical reactions some examples of which were given above), and its outputs are qualitative statements such as

(components of {charcoal} are {phlogiston ash}).

The system's conclusions can be viewed as explanations of the structure of individual substances, and are quite different from the descriptive laws produced by GLAUBER.

6.2.2 STAHL's rules and discovery methods

STAHL has several rules for analyzing chemical reactions and building componential models. These rules are: 'infer-components', 'reduce', 'substitute', 'identify-components' and 'identify-compounds', and are shown in Figure 12 with their interactions with STAHL's knowledge base. The 'infer-components' rule deals with simple synthesis and decomposition reactions, and enables the system to infer the components of a compound. It is stated as

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

Figure 13 shows how this rules operates. To see this rule in action, consider the reaction

(reacts inputs {lime} outputs {quick-lime fixed-air}).

From this reaction, 'infer-components' deduces the components of lime as quick-lime and fixed-air. STAHL's other rules transform complex descriptions into simpler ones, so that they can be matched by the 'infer-components' rule. One such rule is 'reduce', which is responsible for cancelling out substances occurring on both sides of a reaction. 'reduce' is expressed as

reduce: If A occurs on both sides of a reaction, then remove A from the reaction.

The third rule STAHL has is the 'substitute' rule the task of which is to substitute in a reaction, the components of a substance which takes place in the reaction. For example, STAHL may know that

(components of {charcoal} are {phlogiston ash}).
(reacts inputs {calx-of-iron charcoal} outputs {iron ash}).

In this case, the 'substitute' rule would rewrite the second statement as

(reacts inputs {calx-of-iron phlogiston ash} outputs {iron ash}).

The 'reduce' rule would reduce this statement to

(reacts inputs {calx-of-iron phlogiston} outputs {iron}),

and 'infer-components' rule would conclude that the components of iron are calx-of-iron and phlogiston.

The history of chemistry has many cases in which a substance was discovered in two different contexts, originally thought to be two distinct substances, and was eventually identified as a single substance. STAHL's two other rules 'identify-components' and 'identify-compounds' model this form of reasoning. These rules operate independently, and are expressed as follows:

identify-components: If A is composed of B and C and A is composed of B and D, and neither C contains D nor D contains C, then identify C with D.

identify-compounds: If A is composed of C and D and B is composed of C and D, and neither A contains B nor B contains A, then identify A with B.

To illustrate how 'identify-components' work, consider that STAHL knows the following statements (1—3):

(1) (components of {iron} are {calx-of-iron phlogiston}).
(2) (reacts inputs {iron vitriolic-acid water} outputs {vitriol-of-iron inflammable-air water}).
(3) (reacts inputs {calx-of-iron vitriolic-acid water} outputs {vitriol-of-iron water}).

Reduce 'water' from both sides of (2) and (3):

(4) (reacts inputs {iron vitriolic-acid} outputs {vitriol-of-iron inflammable-air}).
(5) (reacts inputs {calx-of-iron vitriolic-acid} outputs {vitriol-of-iron}).

Infer components from (5)

(6) components of {vitriol-of-iron} are {calx-of-iron vitriolic-acid}).

Substitute this in (4) and reduce 'vitriolic-acid' from both sides

(7) (reacts inputs {iron} outputs {calx-of-iron inflammable-air}).

Finally, infer the components of iron,

(8) (components of {iron} are {calx-of-iron inflammable-air}).

Using (1) and (8), STAHL's 'identify-components' rule arrives at the conclusion that phlogiston and inflammable-air are identical.

6.2.3 STAHL's control structure

STAHL processes one reaction at a time, generating as many inferences as possible from it. After

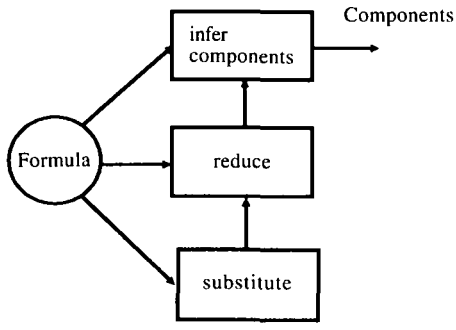

Figure 14 STAHL's control structure in dealing with alternative inference paths.

the system has applied all of its rules to the reaction, it checks the resulting componential models for internal consistency, and for consistency with componential models that have been accumulated from previous analyses. Inconsistencies arise from the violation of the conservation of types of substances. These are caused either by an error in the input to STAHL or an inappropriate application of 'reduce'. If STAHL fails to construct a componential model based on a given reaction, it cancels any intermediate conclusions and remembers the reaction in its original form until new information becomes available. This procedure provides some protection against errors introduced by 'reduce'.

Since STAHL's rules can operate independently there can be many alternative inference paths. To deal with both alternative paths and inconsistencies, STAHL uses a well defined control structure, which is shown in Figure 14. As shown in the figure, STAHL first tries its 'infer-components' rule on a reaction. If this fails, then it tries the 'reduce' rule followed by 'infer-components', or 'substitute' followed by 'reduce' and infer-components'. In any case, the 'infer-components' rule is the last rule to be tried.

If alternative componential models are inferred, STAHL tests each model for additional confirmations. The one that receives more confirmations than others is accepted as the correct model. At each point in its processing, the system's memory contains a list of beliefs about which substances are primitive elements and about the components of complex substances. When all reactions have been considered, some of them may remain un-analysed; in this case, STAHL applies the same cycle to the remaining data recursively. Although STAHL is basically a data driven system at the beginning, it acquires theory driven features as it discovers componential models and makes effective uses of them in further discoveries.

6.2.4 STAHL's theory revision

STAHL is the first discovery system that uses theory revision methods. The system recognizes two types of inconsistency: different componential models for the same substance; and a reduced reaction with inputs but no output or with outputs but no inputs. In such cases, the system recalls all the original reactions that contain substances involved in the contradiction. STAHL then reconsiders the reactions, this time using all the structural knowledge it had already collected, omitting only the inconsistent componential models.

STAHL cannot make generalizations about chemical substances as GLAUBER does. It cannot make generalizations about the reactivities of substances. However, the program could be modified to predict the outputs of a reaction from its inputs, using componential models and its knowledge of reactivities. An integrated system with some of these capabilities has been described in Kocabas (1989, pp. 1984–214).

6.3 STAHLp and REVOLVER

An improved version of STAHL, developed by Rose and Langley (1986), is STAHLp. This system uses an assumption based belief revision method (de Kleer, 1986). It is one of the earliest

computational models of theory revision. The system can even simulate the transition from the phlogiston theory to the oxygen theory in 18th century chemistry, though its simulation is only partial.

STAHLp uses a similar data structure to STAHL's in representing its domain knowledge, and can accept the same inputs, but there is a significant difference in its representation. STAHLp uses source tags in its reaction formulae, and componential models to indicate the premises from which each substance in the reactions and models was derived. These source tags play an important role in the system's theory revision process. STAHLp also has the same inference rules as its predecessor with an important modification on the reduction rule to eliminate certain types of errors in STAHL's reasoning about balancing chemical reactions.

There are significant differences between STAHLp's and STAHL's theory revision methods. Unlike its predecessor, when STAHLp infers an unbalanced null reaction such as $nil = A + B$ from its data, it tries to balance this reaction by a series of operations such as adding A and B to the left, or adding A to the left and deleting B from the right. Each of these changes yield new statements that constitutes STAHLp's 'effect hypotheses' concerning the null reaction. The system next determines what changes in the premises must be made so that they lead to the desired effect hypothesis. These possible modifications are called 'cause hypotheses'. In its quest for cause hypotheses, the system can be viewed as using a form of abductive inference.

STAHLp uses an evaluation function in selecting the best effect hypothesis among the alternatives. Once the best hypothesis is chosen, the system generates a new set of beliefs and models, based on the revised premises. When a contradiction arises, using their source tags the system traces the statements causing the contradiction and revises only those hypotheses that are responsible for the contradiction. In this way, STAHLp can even change its original beliefs as necessary. In resolving one such contradiction involving the oxidation of mercury, the system provides a partial explanation of the transition from the phlogiston theory to the oxygen theory.

REVOLVER (Rose & Langley, 1988) is a variant of STAHLp that uses different theory revision methods. The system was constructed to find models of objects consistent with initial beliefs about them, and to revise these in response to contradictions with new data. Its task domains are particle physics and chemistry. In particle physics it finds the quark models of the elementary particles from their reactions and proposed models by using a set of general heuristics.

REVOLVER uses STAHLp's representation and inference rules with some domain related modifications in its search methods for consistent models. The system employs three heuristics in its theory revision. One involves preferring premises that support a fewer number of beliefs. Another one, which is based on groups, minimizes the complexity of models. The third one increases the chances that each premise leading to an inconsistency will eventually be revised if the search begins to cycle among the same beliefs.

REVOLVER uses a hill-climbing search method in its quest for consistent domain theories. Hill-climbing methods rely on an evaluation function to guide search. However, in this method local peaks do not always lead to the correct solution. Rose (1989) describes the use of a 'minimum revision criterion' to prevent such an effect in REVOLVER's behaviour in finding the quark models of elementary particles.

6.4 BR-3

Scientific theory development has to consider two main problems: inconsistency and incompleteness. Incompleteness means that there are facts within the scope of the theory, which are not explainable, while inconsistency means that certain observations contradict the hypotheses or explanations of the theory. In scientific theory development, incompleteness and inconsistency are usually local problem states as long as they do not involve fundamental theoretical concepts. Resolution of such conflicts depends on their identification in a clear way. In general, incompleteness can be resolved by introducing independent beliefs or hypotheses that can explain previously unexplainable facts. On the other hand, contradictions can be resolved by deleting those beliefs or

hypotheses that contradict the facts or by reducing their generality. The BR-3 program (Kocabas, 1991 a), described below, is capable of identifying and resolving such conflicts within the general strategies just stated. The system models the discoveries of a series of quantum properties and the related conservation laws. In order to explain the program's simulation, it is appropriate to give some background knowledge about its application domain.

6.4.1 The domain of particle physics

Particle physics and elementary chemistry have some similarities. Just as many chemical elements react to produce chemical compounds, the elementary particles (or subatomic particles) interact to produce other such particles. Also, as some chemical compounds decompose into their simpler constituents, most heavier elementary particles decay and produce lighter particles. Similarly, particle reactions are constrained by a set of quantum laws (e.g., see Davies, 1985), as the occurrence of chemical reactions are constrained by a number of chemical laws. These laws dictate the conservation of certain quantum properties in particle reactions. These properties include energy, electrical charge, lepton number, baryon number, spin and strangeness. Table 2 shows some of the well-known particles with their quantum properties and values. The electrical charge conservation law was one of the earliest known laws in quantum physics. This law can be stated as

> *The sum of the electrical charges of the particles entering a reaction is equal to the sum of the charges of the resultant particles.*

Two reactions that conserve electrical charge and that have been 'observed' by physicists are

$$p + p \rightarrow p + n + \pi$$
$$\pi_0 \rightarrow \gamma + \gamma$$

where p, n, π, π_0 and γ resignate the proton, neutron, pion, pion-zero and gamma particles, respectively. However, reactions such as

$$p \rightarrow \pi + \pi_0$$
$$p \rightarrow e + \gamma$$

never happen, despite the fact that they apparently obey the charge conservation law. A theoretical

Table 2 The table of elementary particles and their quantum properties. Symbols: γ: gamma, ν_e: electron-neutrino, ν_μ: muon-neutrino, ν_τ: tauon-neutrino, e: electron, μ: muon, τ: tauon, π: pion, π_0: pion zero, K: kaon, K_0: kaon zero, η: eta, p: proton, n: neutron, Λ: lambda (All antiparticles not shown in this table are designated by "–" over the original particle symbol)

Particle	Mass (MeV)	Charge state	Spin	Lepton number	Baryon number	Strangeness
γ	0	0	1	0	0	0
ν_e	0	0	$\frac{1}{2}$	1	0	0
ν_μ	0	0	$\frac{1}{2}$	1	0	0
ν_τ	0	0	$\frac{1}{2}$	1	0	0
e	0.511	−1	$\frac{1}{2}$	1	0	0
μ	105.66	−1	$\frac{1}{2}$	1	0	0
τ	1784.2	−1	$\frac{1}{2}$	1	0	0
π	139.57	1	0	0	0	0
π_0	134.96	0	0	0	0	0
K	493.67	1	0	0	0	1
K_0	497.67	0	0	0	0	1
η	548.8	0	0	0	0	0
p	938.28	1	$\frac{1}{2}$	0	1	0
n	939.57	0	$\frac{1}{2}$	0	1	0
Λ	1115.6	0	$\frac{1}{2}$	0	1	−1

framework based only on the charge conservation law would be incomplete concerning particle reactions. The discrepancy between the theoretically valid and physically observable reactions was a conflict that had to be resolved.

Physicists resolved such conflicts by postulating new quantum properties and laws so that the physically unobservable reactions were made theoretically invalid by these laws (Omnes, 1970). Their conclusion was that these reactions did not conserve a quantum property, a new kind of 'charge', possessed by some of the particles in the reaction formulae. Thus, they assigned quantum values such as $-1, 0$ and 1 to the elementary particles in such a way that these values are balanced in observed reactions, and unbalanced in those that do not occur. However, such value sets frequently contradicted with certain other observed reactions, and the physicists had to revise them, until they found a consistent set of quantum values. BR-3 simulates the discovery of quantum properties in the same way.

6.4.2 Knowledge representation

BR-3's descriptive knowledge is represented in labelled predicate statements, which fall into three main categories: logical knowledge; formal knowledge; and theoretical, hypothetical or empirical knowledge. Logical knowledge contains the definitions of logical functions and relations such as 'and', 'or', 'for all' and 'if-then-else'. formal knowledge includes definitions, class memberships and class-superclass relationships. Finally, theoretical knowledge includes statements expressing the physical properties of domain objects and certain relationships between them (a more detailed account of categorization of descriptive knowledge can be found in Kocabas, 1989). Examples of BR-3's formal knowledge include the beliefs that

An electron is a particle
A neutron is a particle.
An antiparticle is designated by $\bar{x}$ where x is a particle.

Here, the first two statements express member-class relationships, while the last one gives a definition of 'antiparticle'. Initial theoretical knowledge contains physical data about the elementary particles, namely their electrical charge values in such statements as

The electron's electrical charge is -1.

Theoretical knowledge of BR-3 also includes statements in which particle reactions are represented as ordered pairs of lists as in GLAUBER (Langley et al., 1987). In this case, the elements of the first list are the reactant particles and the second specifies the resultant particles. Valid reactions are identified by the internal label 'reaction', while physically unobserved reactions by the label 'unobserved reaction'.

The system's initial inputs are the basic formal and theoretical knowledge of its domain, a set of observed reactions, and a set of unobserved reactions. Its outputs are a set of new beliefs, categorized as theoretical statements, about postulated properties that can explain the system's previous state of incompleteness and/or inconsistency. Some example outputs are

The qp1 value of p is 0.
The qp1 value of e is 1.

where qp1 indicates the postulated quantum property, and 0 and 1 are the values of the property for p and e, respectively.

BR-3 is an incremental discovery system. Once it reaches a consistent state in terms of its formal and theoretical domain knowledge, new observed and/or unobserved reaction formulae can be given to it for evaluation. These may initiate a new search for a consistent and complete state, leading to further changes in theoretical knowledge.

6.4.3 BR-3's discovery operators and its theory revision

BR-3's theory formation and revision is directed by its drive for completeness and consistency. As

the system tries to achieve a complete and consistent knowledge state, it increases and improves its knowledge. The program identifies two types of incomplete knowledge states: the unexplained absence of an event, and the absence of qualitative or quantitative values of a property for some domain objects. The system resolves the first type of conflict by postulating new domain properties, which in turn are used in new domain hypotheses about the values of these properties for the domain objects. The second type of conflict is resolved by default value assignments, which are revised when they contradict new data. BR-3 identifies three types of contradictions: an unbalanced particle reaction, multiple values of a property for a domain object, and algebraic contradictions (i.e., a linear equation with no solution).

The system revises its theoretical knowledge by using three operators—CHECK-CONSISTENCY, CHECK-COMPLETENESS and REVISE BELIEFS—which control a series of inference rules or procedures. Figure 15 shows the interactions among them with the arrows showing the flow of control. Briefly, the CHECK-CONSISTENCY operator checks for consistency between the quantum values and observed reactions. If the test succeeds, then the operator passes the control to CHECK-COMPLETENESS; otherwise, control goes to the REVISE-BELIEFS operator. In its search for consistency, the program uses a depth-first search constrained by algebraic and domain rules.

CHECK-CONSISTENCY tests the existing valid reactions in the knowledge base against the quantum values of the elementary particles. It decides that a reaction is consistent when, for each quantum property, the corresponding conservation law is satisfied. If some valid reactions violate a quantum law, this means that the system's knowledge of quantum values is in contradiction to the reactions. CHECK-CONSISTENCY labels all such reactions as 'unbalanced', and passes control to REVISE-BELIEFS.

When BR-3 achieves a consistent knowledge state, it activates its CHECK-COMPLETENESS operator, which first checks if there are any unobserved reactions in the knowledge base. If there are none, then this means that, with respect to the quantum values and particle reactions, the system's knowledge is complete. If there are unobserved reactions, the operator checks each unobserved reaction against the quantum conservation laws in order to explain their absence by the violation of one such law. If all the unobserved reactions violate some quantum law, then BR-3's domain knowledge is complete. When the knowledge base is consistent and complete, the system comes to a halt.

On the other hand, if an unobserved reaction is found to be 'valid' by all the quantum laws that BR-3 knows, then the CHECK-COMPLETENESS operator postulates a new quantum property. Then it transforms the unobserved reaction formula into an algebraic inequality to find the sets of quantum values that would make the reaction invalid in terms of the new property. Once the system has added these values to its knowledge base, it must check for their consistency with the observed reactions in the knowledge base, and so passes the control to CHECK-CONSISTENCY.

The REVISE-BELIEFS operator is activated when the system's theoretical knowledge about certain quantum values for some particles are inconsistent with any valid particle reactions. In such cases, REVISE-BELIEFS changes the system's beliefs about such quantum values. After such revisions, the control passes to CHECK-CONSISTENCY. Control is transferred from one operator to another as described above, until the system reaches a consistent and complete knowledge state regarding the particle reactions and quantum properties, or until it is interrupted.

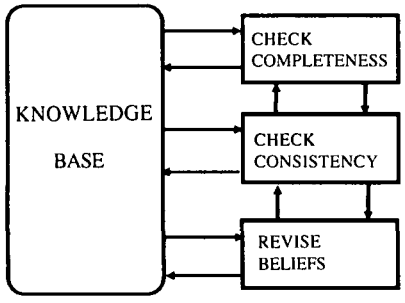

Figure 15 BR-3's discovery operators and their interactions.

The system employs a combination of heuristic search and depth-first search in its theory revision. When BR-3 achieves a consistent knowledge state, it checks for the completeness of its theoretical knowledge concerning domain events, and generates new domain properties and relationships, as necessary. In this way, the program successfully reproduces the discoveries of the lepton, baryon, electron and muon number properties and the corresponding quantum values for the elementary particles by physicists in this century. A detailed description of BR-3's discoveries can be found in Kocabas (1991 a). The system's predictive ability concerning the validity of particle reactions increases as the system develops its theory by formulating new quantum laws upon examining experimental data.

BR-3 differs from earlier systems in its knowledge representation. The system's descriptive and definitive knowledge is represented in labelled predicate statements, in which the labels indicate the category of the statement. The program has a wider scope than the earlier systems in its class, as it combines theory development with theory revision. However, compared with more recent systems such as AbE (O'Rorke et al., 1990) and COAST (Rajamoney, 1990), its knowledge representation is simplistic, as the latter have more structured representations for certain data types such as events and processes.

Despite the generality of some of its methods (see Kocabas, 1991 a), BR-3 operates in a limited domain of physical science. The system lacks the capabilities of performing a number of scientific research tasks such as formulating research goals, setting research frameworks, determining research strategies and methods, experiment designs and planning, and data collection and evaluation. The following subsection describes a program that simulates some of these research activities.

6.5 KEKADA

KEKADA (Kulkarni & Simon, 1988) is a theory driven discovery system which models empirical discovery. It was originally designed to model the discovery of the urea cycle by Hans Krebs in the 1930s, but some of its methods and strategies are general and applicable to other fields in experimental science. Like some earlier systems such as AM, EURISKO and BACON, it relies on a two-space search strategy (Simon & Lea, 1974). According to this model, proposed experiments define the instance space, while the outcome of the experiments determines the rule space (i.e., the hypotheses and rules to be generated). KEKADA's design philosophy was to try to model as many different aspects of scientific discovery as possible and so provide a more comprehensive computational model.

6.5.1 Knowledge representation

The KEKADA system is implemented in the production system language OPS5. Its domain knowledge is represented in frames which contain attribute-value pairs. The system has a set of condition-action rules and a dynamic memory. Thus, KEKADA combines rule-based and frame representation. The system's domain knowledge contains knowledge about chemical processes, substances, experiments, supplementary facts and hypotheses.

6.5.2 KEKADA's operators and their interactions

The program has 64 rules, each of which has a set of conditions and a set of actions. As the AM system, KEKADA uses an agenda of tasks and works in cycles. On every cycle, conditions of each rule are matched against the current state of the working memory. Fired rules may alter the state of the working memory, so that new rules may match the working memory on the next cycle. This process continues until no rules are matched or the system is interrupted.

KEKADA's rules are nominally organized into functional groups called operators. The system has 10 such operators: Problem Generators, Problem Choosers, Hypothesis or Strategy Choosers, Decision Makers, Experiment Proposers, Expectation Setters, Experimenters, Hypothesis Generators, Hypothesis Modifiers and Confidence Modifiers. Interactions of KEKADA's rules are

shown in Figure 16. Accordingly, the system's Problem Generators generate its research problems. This operator has only one rule

> *If the outcome of an experiment violates expectations for it, then make the study of this puzzling phenomenon a task and add it to the agenda.*

Problem choosers choose a problem from the agenda, Strategy Choosers choose a strategy for experimentation, Experiment Proposers suggests experiments, Decision Makers help to choose an experiment, Expectation Setters set the expectations about the current experiment, Experimenters carry out the experiments, Hypothesis. Generators generate hypotheses, and Confidence Modifiers modify the confidences of hypotheses. Any results which contradict the expectations prompt Problem Generators to add the task to the agenda.

6.5.3 KEKADA's background knowledge

The system's domain knowledge is divided into three classes: knowledge about substances, processes and previous experiments. Its knowledge about substances includes the chemical formulae, cost and availability of chemical substances such as amino acids and glucose. The program knows the quantity of a substance used in the experiments. It also has information about the similarities between substances.

The system's knowledge about chemical reactions includes the inputs, outputs, the class of the chemical reaction and supplementary facts about the chemical groups involved in the reaction. Its knowledge of previous experiments provide it to set expectations for the initial experiments. KEKADA also allows the creation of new working memory elements to simulate knowledge acquisition from literature and from experts.

Kulkarni and Simon (1988) developed KEKADA's knowledge base on the basis of Holmes' (1980) account of the discovery of the orhithine cycle by Krebs. Urea was synthesized in the laboratory early in the 19th century and knowledge of its composition and synthesis paths led to certain hypotheses as to how it might be synthesized in living organisms. The authors give a detailed account of the progress of Krebs' research leading to the discovery, including the intermediate hypotheses he may have formed and his cooperation with his colleagues in his research.

6.5.4 KEKADA's discoveries

Kulkarni & Simon (1988) describe KEKADA's simulation in terms of the activities caused by its rules, including the hypotheses generated during the process. Their description contains about 100

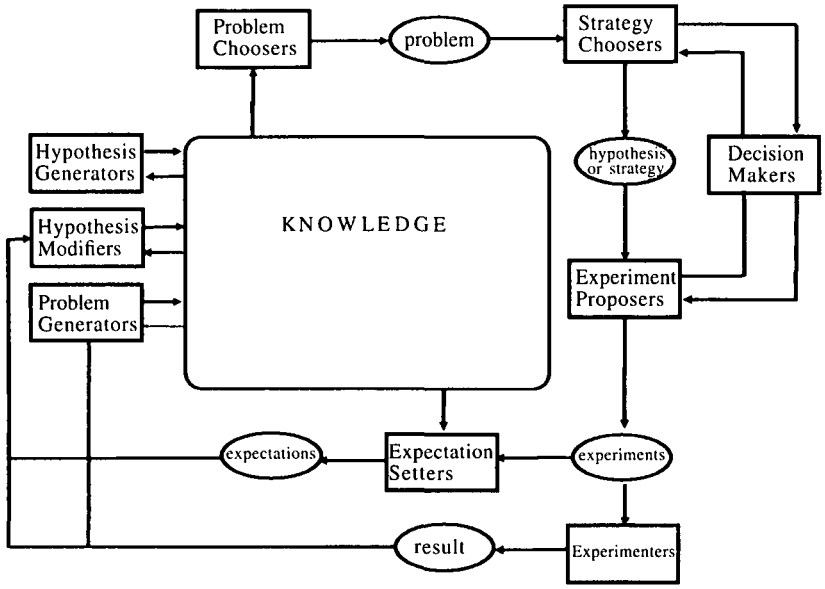

Figure 16 KEKADA's operators and their interactions.

system decision steps, excluding the repetitions (which are about the same number). Some decision steps inevitably involve external (i.e., user) interaction. These include choosing between alternative goals and methods in the absence of historical information, and in supplying the experiment results.

KEKADA has been applied to other areas such as Williamson's discovery that common ether contains two ethyl groups, and Faraday's discovery of the induction of electricity from magnetism (see Kulkarni, 1989). In both cases, a slightly improved version of KEKADA has been used, and the program traces are not as lengthy as in the simulation of the discovery of the urea cycle.

Among KEKADA's 64 action-rules, 31 are domain independent and 33 are domain specific. The domain independent rules are claimed to be used by scientists in various deciplines. This claim is substantiated by the CER system (Kocabas, 1989) by its use of some of KEKADA's rules, albeit in modified or more precisely expressed forms. Some examples of these general rules are

If the cost of a substance to be tested is too high, eliminate it.
If a substance to be tested is not available, eliminate it.
If the goal is to study a reaction in detail, carry out the reaction under various conditions.

KEKADA's rules on the choice of raw materials and processes can be said to be general. However, some of its rules of strategy choice are not clear, and in certain cases confused with its Experiment Proposers and with the other rules of the system. Despite its shortcomings, the system constitutes a significant attempt to model different aspects of scientific discovery in an integrated way.

6.6 AbE and COAST

Recent discovery systems are increasingly using methods that allow more structured and fine grained representations of domain knowledge. Two such systems are AbE (O'Rorke et al., 1990) and COAST (Rajamoney, 1990).

6.6.1 AbE

O'Rorke et al. (1990) propose abduction as a general method for theory formation and revision. They argue that an observation contradicting a prediction of a given theory can be explained in terms of the basic principles of the theory, claiming that this process can lead to new hypotheses by abductive inference. They also describe AbE, a system which uses this strategy for theory formation and revision.

The primary task domain of AbE is 18th century chemistry and its two conflicting theories: the phlogiston theory and the oxygen theory. These theories would 'explain' a series of chemical processes such as combustion, oxidation and calcination. AbE simulates the stages of the transition from the phlogiston theory to the oxygen theory. In other words, it models a paradigm shift in its domain.

AbE represents chemical processes in qualitative schemas (Forbus, 1984). Each schema describes a process in terms of the substances involved, the preconditions for the process to take place, the process conditions, and a set of relationships that indicate the qualitative changes in the process. In this way, the assumptions of a theory are also captured in the representation. For example, the phlogiston theorists' belief that all combustibles are compounds containing the element phlogiston captured in the 'process conditions' slot. The basic laws of AbE's domain theory are represented in simple and complex predicate statements. One such law is: 'Combustion decreases the amount of phlogiston in charcoal'. The inputs to AbE are its domain knowledge which consists of process descriptions and the laws of the domain theory (including the general laws of qualitative physics). The latter are represented as predicate statements. The system's output is a set of explanations and a revised theory.

AbE's basic method of discovery is abductive inference on the qualitative descriptions of domain processes. The system employs a best-first search method to construct explanation trees to account for an observation. When an observation contradicts its domain theory, AbE deletes those

hypotheses that are responsible for the contradiction, and attempts to construct the explanation tree for the observation. When it fails to do so, it uses its basic knowledge of domain theory (physics) to complete the explanation by introducing new hypotheses. In this way, it arrives at a partially modified theory. As these modifications take place in an incremental fashion, one domain theory (e.g., phlogiston theory) gradually transforms into another (e.g., oxygen theory) that is consistent with new observations as well as the old ones.

However, not all theory formation and revision processes can be accounted for by abductive inference only. Some involve a combination of abstraction and abduction (see Kocabas, 1991 a). AbE's new hypotheses generated by abductive inference are derived from its basic knowledge of domain theory, whereas an abstraction expands a theory beyond a system's basic knowledge. On the other hand, it should be noted that AbE carries abduction over more structured and detailed representations than the earlier systems.

6.2.2 COAST

Rajamoney (1990) describes COAST, a system that employs a knowledge-intensive theory revision method. This system has been tested on problems from physics that involve a variety of processes such as evaporation and heat transfer.

COAST's theoretical knowledge is represented in qualitative process schemas as in the AbE system. Domain objects are described using similar data structures. Physical systems (or "scenarios") are represented in frames with layout and behaviour components. The former specifies the objects of the system, their arrangement, and the initial physical relationships between them. The latter represents the qualitative changes in the physical state of the objects.

The inputs of COAST are the observed changes in the physical system under study, for which it tries to construct explanations. When the program fails to explain an observation, it revises its domain theory. The first step in this process involves identifying three different types of failures: incomplete explanations, contradictions and mutually inconsistent multiple explanations. In general terms, COASTS's identification of failures has similarities to STAHLp's and BR-3's identification of conflicts.

The theory revision methods of COAST focus on making partial changes in the description of the physical system behaviour (or scenario), such that the new scenario is consistent with the current set of observations. The program generates a number of alternative theories during its revision process. Rajamoney (1990) defines theory development as a complex and continuous process that involves three major activities: theory formation, theory revision, and paradigm shifts. His program combines theory revision and paradigm shifts as the AbE system.

COAST designs experiments to test and evaluate its alternative theories to select the best one. The system uses general heuristic methods to constrain search in theory revision. The program generates representative partial models (or "exemplars") of its domain knowledge, and tests its proposed theories against them for validity. The remaining theories are examined for their ratings in terms of their structural and explanatory simplicity and predictive power using a small set of metrics. The system's use of partial models to guide search constitutes a clear improvement on the earlier systems. Also, its ability to design experiments to constrain search for alternative theories is a useful feature that most of the earlier systems lack. COAST's theory revision methods are also more general than those of the earlier systems (with the exception AbE, as this is based on making partial changes in the process descriptions in both systems).

7 Quantitative discovery: BACON and IDS

In this section we shall examine two systems: BACON (Langley, 1978; 1990) and IDS (Nordhausen & Langley, 1987). The first one constitutes a landmark in computational modelling of scientific discovery, as it is the first detailed and successful attempt to simulate quantitative discovery. The second program is equally important in its own right, because it integrates qualitative and quantitative discovery methods.

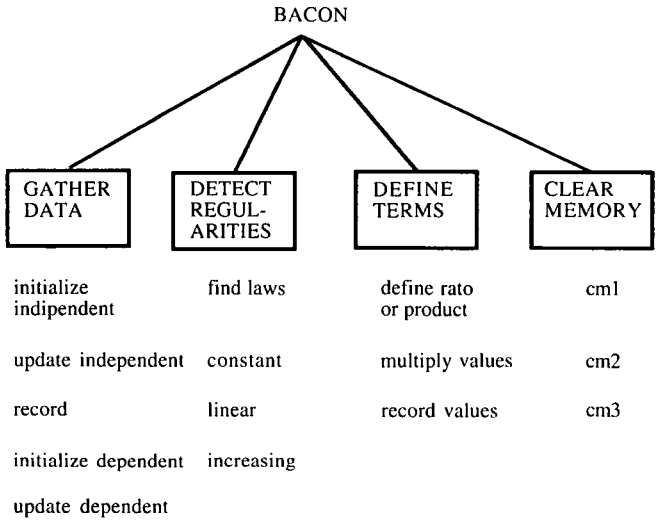

Figure 17 BACON's tasks and rules of action.

7.1 BACON

BACON (Langley et al., 1987) is a series of data driven discovery systems (BACON1 to BACON6) which model quantitative discoveries in physical sciences. BACON1 (Langley, 1978) is the earliest and the simplest of the series, which nevertheless has the ability to rediscover an impressive set of physical laws such as Boyle's Law, Kepler's Third Law, Galileo's Law and Ohm's Law. The subsequent versions of BACON model the discoveries of more complex laws as well as the intrinsic properties of substances. They also have a few additional amendments on the rules and techniques of BACON1, but the basic functions of these systems are the same. In the remaining part of this study we refer to them as BACON, stating the differences where necessary.

The main emphasis in BACON's design has been to the generality of the system. It has a small set of rules, which has proved to be successfully applicable to data from a wide range of physical systems. For this reason, the BACON system can be considered as a general problem solver. Its search has similarities with the two space search model of Simon and Lea (1974) in the way it searches a data space and a rule space, and attempts to relate one to the other.

7.1.1 Knowledge representation

BACON's experimental data are represented in lists which are sets of attribute-value (or variable-value) pairs linked to a common node. These represent a series of observations that have occurred together. The system has two types of variables: independent and dependent. It has control over independent variables; it can vary the values and request the corresponding values of the dependent variables. Independent variables can take either numerical or symbolic values. When BACON defines a new term, it generates a name for it and treats it in the same way as its other terms. Defined terms can be used in defining other new terms. Some of these terms are dispensed with when BACON completes its discovery of a particular law and expresses it by the original terms.

BACON's heurists are represented in a production-system, which consists of a set of condition-action rules using a dynamic working memory. The system operates in cycles. On every cycle, the conditions of each rule are matched against the current state of the working memory. When a rule is successful, its actions affect the state of the working memory, making new productions match.

7.1.2 BACON's control structure

BACON's 16 rules are divided into four groups (see Figure 17) to perform the following tasks: data gathering, detecting regularities, defining terms and cleaning up the memory when certain stored elements are no longer useful. The data gathering operation uses five rules: 'initialize-

independent', 'update-independent', 'record', 'initialize-dependent' and 'update-dependent'. The first of these rules initializes the values of the independent variables of the physical system, and sets BACON to iterate through these values. The second rule asks for the values for the current state and iterates through the remaining values of the independent variable. The third rule receives and records the values of the dependent term. The fourth creates an initial list for the values for the dependent term. Finally, the fifth rule adds the values of the dependent term to that list when they are received. In order to be able to find the quantitative relationships between terms, the program needs at least three sets of values for each state.

After the data gathering is complete, BACON's second and most important task is to discover the regularities in the data. This task is carried out by a set of five rules. The first rule, 'find-laws', checks the iterations through the independent variable and when the iteration is complete, it tries to find laws for the dependent variables. The second rule, 'constant', tries to find if the dependent term has the same value in all data clusters. If this is the case, it infers that the dependent term has a constant value. The third rule, 'linear', tries to find if the relationship between the two variables is a linear one. If this is the case, it finds the slope and the intercept of the linear relationship. The fourth rule, 'increasing', tries to find if the values of both variables are increasing. If this is the case, then it recommends BACON to consider the ratio of the two variables. The last rule, 'decreasing', tries to establish if the value of one of the variables is increasing while the other is decreasing. If this happens, it recommends BACON to consider the product of the two variables.

The third task of BACON is carried out by three rules: 'define-ratio-or-product', 'multiply-values' and 'record-values'. The first rule defines the ratio or the product of the two variables according to the message coming from the rules 'increasing' or 'decreasing'. The 'multiply-values' rule multiplies the current values of the defined term with the values of the independent variable. The 'record-values' rule records the current values of the defined term. Once these values have been recorded, BACON's rules for detecting regularities can look for constant values and relations to other terms, and when all the relationships are established between the terms, the 'find-laws' rule formulates the law discovered.

The later versions of BACON (e.g., BACON3) use a differencing technique to find laws. This technique is more general and enables the system to discover polynomial relationships between domain variables. All versions of BACON have a mechanism to deal with noisy data, which consists of using a maximum percentage deviation, which functions as a measure of data reliability. BACON1 was designed to deal with physical systems with only two variables. Later versions can cope with more complex physical systems. BACON3 designs its experiments in such a way that by holding some variables constant, only two terms are compared at a time. When the comparisons are completed, other variables are set free one by one, until all the binary relationships are found. Then these relationships are combined to formulate the law. In the following two subsections, some of the discoveries of BACON using a Prolog version of BACON3 instead of the original program are described.

7.1.3 BACON's rules of discovery

BACON uses a simple but general set of rules in its search for quantitative discoveries. These rules concern operations on the values of variables in a physical system, and are expressed by Langley et al. (1987) as follows:

If the values of a term are constant, than infer that the term always has that value.
If the values of two numerical terms increase together, then consider their ratio.
If the values of one term increase as those of another decrease, then consider their product.

To illustrate BACON's simulation of Kepler's discovery of his third law by using these rules, consider the data given in Table 3, about the planets Mercury, Venus and Mars. The second column contains the values of the sideral periods of the planets in days, and the third column their distances from the sun.

When BACON is run, it asks for the names of the domain variables, which are specified by the

user, say, as P and D. Then it asks the names of the 'test-cases', which are the names of the planets. Then for each planet, the values of the first variable, P, are collected. The same process is repeated for the second variable, D. BACON then asks for the maximum deviation from the mean value of the constant term, which is usually a few percent. In this case it is chosen to be as 5%.

When BACON considers the data, it notices that the values of the variables are not constant. So the 'constant' rule fails to match with the data. Similarly, the 'linear' rule also fails, but BACON detects that the values of both variables are increasing. This activates the 'increasing' rule, which proposes BACON to consider the ratio D/P of the two variables. The 'define-ratio-or-product' rule defines the ratio as $T1$ and the definition $T1 = D/P$ is recorded. The new term is added to the top of the list of the term which includes D and P. The 'record-values' rule records the values of $T1$ as [0.659, 0.483, 0.330].

BACON tries to find regularities between the values of $T1$ and D. The 'constant' rule fails to match. Similarly, the 'linear' and the 'increasing' rules fail, but the 'decreasing' rule is activated because the values of $T1$ decrease while those of D increase. So, this rule proposes BACON to consider the products of $T1$ and D. The 'define-ratio-or-product' rule defines the product $T1.D$, and the definition of $T2 = D^2/P$ is recorded. The new term is added to the top of the list of terms. The 'record-values' rule records the values of $T2$ as [38.23, 51.84, 75.01].

BACON tries to find regularities between $T2$, D and P. The 'constant' and 'linear' rules fail, but 'increasing' succeeds and the system considers the ratio of $T2$ and P. It ignores the ratio of $T2$ and D, because this would be equivalent to $T1$ which was previously defined as D/P. The 'define-ratio-or-product' rule defines the ratio $T2/D$ and the definition $T3 = D^2/P^2$ is recorded. Again, the new term is added to the top of the list of terms and the values of $T3$ are recorded as [0.434, 0.230, 0.109].

BACON now tries to find regularities between $T3$, D and P. This time it considers the product of $T3$ and D, for D is increasing and $T3$ is decreasing, and the product is not equivalent to a term generated earlier. The 'define-ratio-or-product' rule defines the product $T3.D$ and the definition $T4 = D^3/P^2$ is recorded. The new term $T4$ is added to the top of the list of terms and its values are recorded by 'record-values' as [25.195, 24.883, 24.783]. When 'find-laws' iterate, BACON notices that the values of $T4$ are constant within the limits of the 0.05 deviation assigned earlier. The 'constant' rule matches with the data on $T4$ and reports $T4$ as constant and records its average value as $T4 = 24.95$. Since the last term is found constant, the comparison of the variables is complete and the current goal of BACON is accomplished. The 'find-laws' rule formulates the law as $T4 = D^3/P^2$ and that $T4$ is constant and is equal to 24.95, and reports the termination of the research activities.

7.1.4 BACON's simulation of the discovery of the ideal gas law

As has been briefly mentioned, BACON deals with complex systems by planning its data gathering. In trying to find the mathematical relationships in systems which have more than two variables, it accepts only two variables at a time while others are maintained constant. In this way, the system finds all the binary relationships between the variables and then combines them by substitution to formulate the general law. The summary of BACON's simulation of the discovery of a complex law (i.e., the ideal-gas laws) is as follows:

The domain variables V (volume), P (pressure), t (temperature) and N (the number of moles) is given to BACON by interaction, as described before. The system chooses two variables from the top of the list, namely V and P, and keeps the other two variables (t and N) constant. Then it asks

Table 3 Data used in the rediscovery of Kepler's third law of planetary motion

Planet	P (days)	D (10 m km.)	T1 (D/P)	T2	T3	T4
Mercury	88	58	0.659	38.23	0.434	25.195
Venus	225	108	0.483	51.84	0.230	24.883
Mars	687	227	0.330	75.01	0.109	24.783

the names of the 'test-cases', which are the names of the tests (e.g., test-1, test-2, test-3). Then for each test-case, the values of the first variable V are collected. The same process is repeated for the dependent variable P. (The values given to BACON are shown in Table 4.) BACON then asks for the maximum percentage of deviation for the constant term. In this case it is chosen to be 1.5%.

After this data gathering, BACON tries to find regularities in the data by using its four rules for the task. The 'constant' rule fails to match with the data. Similarly, the 'linear' and the 'increasing' rules also fail, but the program detects that the values of V are decreasing while those of P are increasing. This activates the 'decreasing' rule and the product of P and V is considered. The 'define-ratio-or-product' rule defines the product $P.V$ as a and the definition $a = PV$ is recorded. Then the 'record-values' rule records the values of a as [2354.5, 2354.5, 2354.5], as shown in Table 4. When the 'find-laws' rule iterates, BACON notices that the values of a are constant within the assigned deviation limits. The 'constant' rule matches with the data on a and reports that a is constant, and its mean value is recorded as 2354.5.

The new term is added to the top of the list of the variables which now includes a, t and N. BACON proposes the variable N to be kept constant and asks for the values of a against t. Arbitrarily, the same number of test-cases are used for the values of a as in Table 5. A data deviation limit of 2% is given.

After the data gathering, BACON tries to find regularities between t and a. The 'constant' rule fails, but the 'linear' rule is activated because the differences between the values of a are [83.3, 83.2], which are considered as constant within the deviation value of 2%. The 'linear' rule reports the linearity between t and a, and tries to find the slope and the intercept of the relationship. It finds the slope between t and a as 0.12012 and the new term is defined as b. The 'linear' rule then finds the slope between t and a as -272.82 and the term is defined as c. BACON's 'find-laws' rule iterates and formulates the relationships between t and a as $t = 0.12012a - 272.82$. The new terms b and c are added to the list of variables which now has b, c and N. BACON's data gathering rules ask for the values of b against N, and the values shown in Table 6 are given to it. A deviation limit of 3% is also given.

BACON tries to find regularities between b and N. The 'constant', 'linear' and the 'increasing' rules fail, but the 'decreasing' rule is activated, for the values of b are decreasing while those of N are increasing. BACON considers the product of b and N and defines a new term d as $d = b.N$. It finds the value of d as 0.12012. BACON then compares the given values of c and N and notices that the values of c are constant. So the 'constant' rule is activated, but since the values of c are constant, it does not define a new term. Instead, it establishes the value of c as $c = -273$ and simply records $c = c$.

BACON's 'find-laws' rule iterates, but by now there are no variables left in the list of variables to consider. Therefore, the 'find-laws' rule reports the termination of the research activity and displays the relationships found as

(6) $c = c,\ c = -273$
(5) $d = N.b,\ d = 0.12012$
(4) $c = t - b.a$
(3) $b = (t - c)/a$
(2) $t = b.a+c$
(1) $a = P.V$

Table 4 Simulated data used in the rediscovery of the ideal-gas law \

N(moles)	t (C)	P (dm-water)	V (lt)	$a = PV$
1	10	101.4	23.22	2354.5
1	10	202.8	11.61	2354.5
1	10	304.2	7.74	2354.5

Table 5 Second-stage data used in the rediscovery of the ideal-gas law

N (moles)	t (C)	$a = PV$	b	c
1	10	2354.5	0.12012	−273
1	20	2437.8	0.12012	−273
1	30	2521.0	0.12012	−273
2	10	4709.0	0.24024	−546
2	20	4875.6	0.24024	−546
2	30	5042.0	0.24024	−546
3	10	7063.5	0.36036	−819
3	20	7313.4	0.36036	−819
3	30	7563.0	0.36036	−819

By combining equation (1) or (2) or (3) or (4) with (5) and (6), the ideal-gas law $PV = 8.32N(t + 273)$ is found. In this way, BACON also introduces the absolute temperature scale $(t + 273)$ for the ideal-gas law, which is expressed more concisely as $PV = 8.32NT$ where T is $t + 273$.

BACON constitutes an important development in the emerging discipline of computational modelling of scientific discovery. Its methods should not be confused with linear modelling, for the system uses a small set of simple heuristics rather than algebraic methods. Additionally, as we have seen in the rediscovery of the ideal gas law, the program has the ability to 'schedule' its experimentation and data collection when dealing with complex physical systems (i.e., systems with more than two variables).

The system has been further developed (to BACON5, BACON6) to simulate the discoveries of the intrinsic properties of substances, such as specific heat and electrical resistivity (see Langley et al., 1987). As it stands, the original program cannot discover trigonometric and logarithmic relationships. However, the program could easily be given these abilities by adding two more condition-action rules. The only problem would be to decide about the order of the rules, as, e.g., the same problem state may satisfy both the 'increasing' and the logarithmic rule. Another shortcoming of BACON is its inability to discover relationships that can be expressed by discrete functions, for this would require more structured representation (e.g., qualitative process schemas) of domain events. The next subsection describes IDS, a program that integrates BACON's quantitative methods with qualitative discovery.

7.2 IDS

IDS (Nordhausen & Langley, 1987; 1990) is an integrated discovery system that formulates both qualitative laws and discovers quantitative relationships. The system detects the qualitative changes in a physical environment by using its sensors, and represents such changes in a structured way to guide its search for quantitative relationships. IDS integrates three aspects of theory formation based on empirical discovery, namely, taxonomy formation, generation of qualitative laws, and the discovery of quantitative laws.

Table 6 Third-stage data used in the rediscovery of the ideal-gas law

N (moles)	b	c	$d = bN$	$c = c$
1	0.12012	−273	0.12012	−273
2	0.06006	−273	0.12012	−273
3	0.04004	−273	0.12012	−273

7.2.1 Knowledge representation

The system represents its descriptive domain knowledge in data structures similar to Forbus's (1984) qualitative schemas. Accordingly, the schemas describe the behaviour of a physical system over time. Each schema consists of a series of state descriptions. State descriptions are linked together in high level relationships, which also include the conditions that must be satisfied in transition from one state to its successor. The program represents each frame with four slots. The 'object description' slot includes physical descriptions of the objects. The 'structural description' slot represents the spatial relationships of the objects in the state. The 'changes' slot contains changes that are taking place in the physical state, indicating an increase or decrease in the values of physical system variables by derivatives. Finally, the 'quantity' slot contains statements about the objects in relation to a set of physical variables such as mass and temperature.

IDS organizes qualitative states into a taxonomic hierarchy in the form of a tree, so that no specific state can belong to more than one category. This taxonomy connects qualitative states at different levels of abstraction through 'is-a' links, but it also contains temporal information. In this way, the system also specifies 'successor' links between states indicating a temporal order between them. The successor relations between abstract state also represent qualitative laws such as those found by GLAUBER (Langley et al., 1987). The system discovers quantitative laws by examining the quantitative changes between the initial and terminal states. These laws are stored in links that connect these states. So, unlike BACON (Langley et al., 1987) IDS discovers quantitative laws within a qualitative context.

There is a significant difference between the construction of qualitative schemas by IDS and other systems. The former constructs such schemas directly from data obtained by its sensors, rather than deducing them from externally provided process descriptions. In addition to its sensors, the system also has a set of 'effectors' by which it can induce changes to its environment, such as moving objects and heating them.

Each of IDS' effectors have a blank qualitative schema. By inducing changes to the physical system with its effectors, and by gathering information from it by its sensors, the program fills in the slots in its state descriptions, and in this way builds up its knowledge. When IDS encounters unfamiliar behaviour, it adds a new state description to the schema.

7.2.2 Methods of discovery

IDS forms its theories by integrating taxonomy formation, search for qualitative laws, and search for quantitative relations between the qualitative states in its knowledge base. In taxonomy formation, the system processes one qualitative state at a time using a hill-climbing approach, and sorts it through its current taxonomy and incorporates it in the hierarchy, and creates new nodes (e.g., its successor) as necessary. In contrast to Lenat's (1978) AM system which organizes its concepts into a hierarchy and dynamically extends it by formal methods, IDS builds its concept hierarchy through observations.

IDS's discovery of qualitative laws is quite different from the way GLAUBER (Langley et al., 1987) discovers such laws, for the program uses its specific qualitative state descriptions and the links between them to hypothesize links between more abstract states. These hypothesized links represent the qualitative laws in varying degrees of generality. In this way, the system is able to represent qualitative laws at different levels of abstraction. Figure 18 summarizes the system's abstraction of the acid–base reaction law.

In its discovery of quantitative laws, IDS uses its domain knowledge represented in qualitative process schemas to guide the search for quantitative relationships between the physical system variables. The program examines the effects of the changes in one of the variables over the others in very much the same way as BACON does, and formulates the numeric and abstract relationships between the variables. IDS's formulation of laws is directly related to the process schemas, such that quantitative terms are specified by subscripts relating to the object and the physical state description.

IDS makes extensive use of its well structured knowledge representation and organization in its

qualitative and quantitative discoveries. However, its advantage is not without drawbacks. Its formulation of qualitative and quantitative laws heavily relies on the taxonomy of qualitative states formed by the system through its observations, and this in turn depends on order of the data received by the program.

The advantages of representing physical process descriptions in qualitative schemas is twofold. First, they provide a qualitative framework in which the quantitative relationships are meaningful. Second, they constrain the search for quantitative laws and enable the system to design experiments in a systematic way. The system uses data-driven reasoning in its search for quantitative relationships. In this it operates like the BACON system. However, its qualitative schema representation provides IDS with the ability to detect new types of intrinsic properties such as the melting points of solids, which the BACON system is incapable of.

IDS constitutes an important step in the computational modelling of scientific discovery by its integration of research activities such as experiment design through controlling physical system variables, data collection through direct measurement, and theory formation through building taxonomies of domain events, and qualitative and quantitative discovery. However, the system lacks an important component of theory development: the current version of IDS has no theory revision capability. Some simple theory revision methods such as hypothesis specialization and deletion based on deleting the links between qualitative states could be amended to it, but this would involve a revision of system's states of knowledge organization.

8 Summary and conclusions

After this survey on some of the prominent computational models of scientific discovery, I can now summarize my observations about the methods used in these systems, and their scope as computational models of scientific discovery. Most of the systems have essentially a rule-based representation (DENDRAL, Meta-Dentral, MOLGEN, GLAUBER, NGLAUBER, STAHL, STAHLp and BACON). All of the systems that use frame representation (GENSIM/HYPGENE, AM, EURISKO, PI, GALILEO, KEKADA, AbE and COAST) incorporate rules. Significantly, some of the more recent systems (IDS, GENSIM/HYPGENE, AbE and COAST) employ qualitative schemas to represent qualitative processes.

The effectiveness of a particular representation depends on the complexity of knowledge and the methods of learning and discovery employed by a knowledge system. Frame representation facilitates formal reasoning. This is a feature that has proved its usefulness in systems such as AM and EURISKO. Rule-based representation is efficient in narrow domains where knowledge can be

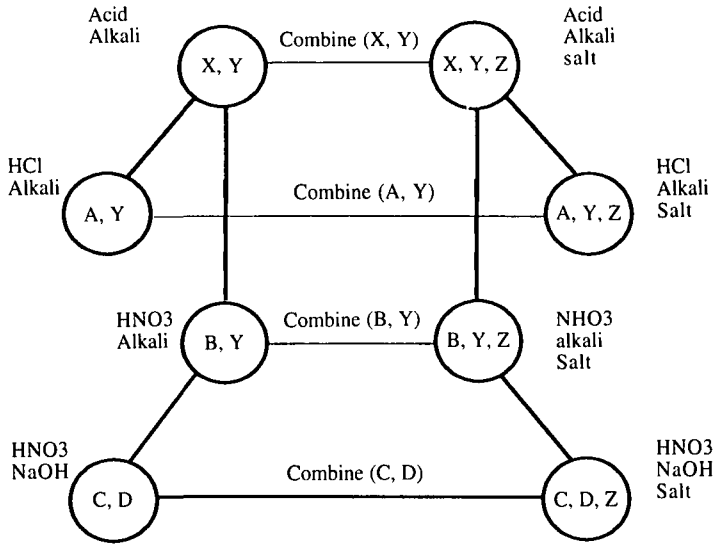

Figure 18 IDS's rediscovery of the acid–base reaction principle through establishing links between the qualitative states of different levels of abstraction.

divided into descriptive statements (i.e., "facts") and rules. On the other hand, qualitative schemas are essential in reasoning about events and processes. It seems that comprehensive models of discovery must be capable of integrating all these representation methods. Lenat (1983 c) suggests that complex knowledge systems must be capable of using multiple representations and shift from one representation to another as necessary.

Considering the close relationship between knowledge representation and learning, the published papers on some of the systems examined in this article do not provide detailed examples of their representation, while in general, the learning and discovery methods are explained well. It was reasonably easy to build some of them (e.g., GLAUBER and BACON) from their descriptions. There was even a criticism on one of the systems, namely AM (Lenat, 1979), for lack of clarity in its representation and learning methods (see Ritchie & Hanna, 1986). However, this situation was later rectified by Lenat (1983 a).

The earlier computational models had limited and well defined tasks. For example, the DENDRAL system has one main task: to determine the molecular structure of an unknown compound from the data from its mass spectrograph. Meta-DENDRAL's task was to generate rules which correlate mass spectrometric data with substructural features of the molecules by comparing the data with the chemical structure of the known compounds. From the standpoint of representation, DENDRAL's and Meta-DENDRAL's input data are uniform. Their domain knowledge is represented in list structures in attribute-value pairs and in inference and condition-action rules. While DENDRAL is essentially a problem solving system within a well developed domain theory, Meta-DENDRAL has the characteristics of a theory formation system.

MOLGEN models an important but rather narrow aspect of scientific research: building hierarchic plans for experimentation. Its domain knowledge is represented in frames where slots contain information about molecules. The program's plans are also represented in frames. The system's methods can be expected to find place in more comprehensive models of the future. In fact, some of its methods (e.g., its rules about the choice of methods and experiment materials) have later been incorporated in KEKADA by Kulkarni and Simon (1988).

GENSIM/HYPGENE's knowledge is represented in process schemas and a taxonomic hierarchy of frames. Significantly, the system stores its domain knowledge in three different knowledge bases, separating formal knowledge, descriptions of experiments, and the information about the qualitative processes of its domain. The program integrates planning methods with theory revision. GENSIM/HYPGENE identifies contradictions between its predictions and the outcomes of experiments. Its theory revision essentially consists of making changes in the process descriptions until it achieves a consistent knowledge state, so that the process schemas are consistent with the observations.

AM and EURISKO uses an agenda mechanism to control the activities of their heuristics over their domain concepts. The main task of AM is to generate new concepts, evaluate their interestingness and discover conjectures about its concepts. On the other hand, EURISKO tackles one of the most difficult tasks in artificial intelligence, the discovery of domain independent rules. However, both AM and EURISKO operate on basically formal knowledge. Lenat's later project. CYC (e.g., see Lenat et al., 1986; Lenat & Feigenbaum, 1987; Lenat & Guha, 1989), is an attempt to capture all types of knowledge (commonsense, technical, formal, empirical and theoretical knolwledge) in a frame-based system built around 'concepts'. However, as can be seen from the examples provided by Lenat (1979; 1983 a) and Lenat et al. (1986), the arrangement of slots in frame-based systems like AM, EURISKO and CYC do not follow an established methodology. The absence of a methodology for organizing knowledge is bound to cause problems in knowledge acquisition, especially during the development stage of complex systems.

PI and GALILEO conduct conceptual and theoretical analysis and synthesis in a domain theory. The former's knowledge is represented in frames and rules, while the latter system uses process diagrams, lists and rules in its representation. Both systems focus on a specific but less studied and important aspect of theory development. Therefore, their methods are expected to be incorporated in any comprehensive model of discovery.

GLAUBER, NGLAUBER, STAHL, STAHLp and REVOLVER's knowledge is represented in predicate statements with labelled arguments. Domain knowledge of these systems includes componential models and reaction formulae. GLAUBER's formal knowledge about clusters constitutes a separate category from its empirical knowledge about chemical reactions, but this distinction has not been made clear by Langley et al. (1987). GLAUBER and NGLAUBER's main tasks are to form classes (clusters) of their domain objects (chemical substances) and build hypotheses by generalizing their individual variables to relevant classes. The STAHL and STAHLp systems have a more varied set of tasks, which are, inferring components, reducing chemical formulae into simpler forms, substitution, identification of compounds and components, and belief revision. The rules of these systems are independent and can, in principle, operate in parallel.

As far as its knowledge representation methods are concerned, BR-3 constitutes a class of its own, as its descriptive knowledge is divided into three main functional categories as logical, formal, and theoretical knowledge. In each category, knowledge is represented as labelled predicate statements where the labels indicate the category of the object level statements. The categorization provides clarity in logical and extralogical reasoning. On the other hand, the system's representation does not include qualitative processes. BR-3 integrates the methods of theory formation and theory revision, albeit in a narrow domain of physical science. The system identifies not only inconsistent, but also incomplete knowledge states as conflicts, and resolves such conflicts by a combination of abstraction and abductive inference.

The KEKADA system (Kulkarni & Simon, 1988) is an important attempt to model scientific discovery as the product of several different series of activities. This system combines rule-based and frame-based representation. Its descriptive knowledge includes formal, theoretical and factual knowledge about chemical processes, substances and experiments, but no categorical distinction has been made between them. KEKADA models the discovery of the urea cycle in biochemistry. The program's tasks are carried out by a well defined set of operators. This organization enables the system to conduct a series of different tasks such as generating scientific research problems to focus on, problem choosing, strategy choosing, proposing experiments, setting expectations about its experiments, generating hypotheses and testing hypotheses. Although the system's design includes a variety of research activities, some of these activities are carried out with user interaction.

AbE's descriptive domain knowledge is represented in frames and schemas. Process schemas provide structured representation of events, and this facilitates reasoning about continuous changes in physical systems. AbE's theory revision is based on abductive inference over qualitative descriptions of domain processes. The program revises its theory by deleting those hypotheses that contradict with observations, and builds new explanations from its basic domain knowledge. The incremental changes in AbE's domain theory gradually transforms it into another theory consistent with observations. In this way, AbE models theory revision and paradigm shifts.

COAST's domain knowledge is also represented in qualitative schemas. Physical systems and objects are represented in structured components of frames. The program designs experiments to test and evaluate its theories. Its theory revision is based on making partial changes in its qualitative process descriptions. The program generates partial models of its domain knowledge to constrain search in theory revision. Like the AbE system, COAST simulates a limited part of scientific research, namely theory revision and paradigm shifts.

Most of the systems we looked at were qualitative discovery programs. The only quantitative discovery systems reviewed were BACON and IDS. The former's knowledge is represented as lists which are sets of attribute-value pairs. BACON is a data driven system and uses three different types of expressions for its domain knowledge: its empirical data represented in lists; its formal knowledge which contains the definitions of new terms; and its hypotheses which represent the mathematical relationships between its domain variables in mathematical formulae. Langley (1978) and Langley et al. (1987) do not make categorical distinctions between these different types of knowledge. However, BACON's tasks related with these types are limited, and do not necessitate such functional distinctions. Although it has a limited scope in the spectrum of scientific

research activities, the system's methods of quantitative discovery are quite general. BACON could easily be integrated with a categorized logic-based knowledge system such as CER (Kocabas, 1989), so that, instead of relying on fed-in data, it can operate on the factual and empirical knowledge of the host system to discover binary quantitative relationships between descriptive domain concepts.

IDS constitutes an important development in machine discovery in more than one respect. It integrates quantitative methods of discovery with qualitative discovery and taxonomy formation. The system has greater autonomy than all the others described, in the sense that it can induce changes to its environment by its effectors, and can gather data directly from its environment by its sensors. In this way, the system constitutes a clear step forward towards building more integrated and autonomous discovery systems.

Very few of the current models have actually made any real discoveries. Some of Meta-DENDRAL's discoveries of domain heuristics in organic chemical analysis (see Buchanan & Feigenbaum, 1978), and some of AM's mathematical discoveries (see Lenat, 1983 a) can be regarded as real discoveries. However, computational modelling of scientific discovery is a very young but active research field in artificial intelligence, and is currently more concerned with understanding the phenomenon than making spectacular discoveries. I think that there is a general consensus in machine learning community that in order to make significant discoveries, a computational model must have the following:

1 An effective knowledge organization which can incorporate different representations, and can transform knowledge from one representation to another as necessary, as different representations facilitate the implementation of different learning and discovery methods.
2 A large amount of well organized domain knowledge, together with general scientific, technical and commonsense knowledge. General knowledge facilitates analogical reasoning which is important in scientific discovery (e.g., see Lenat & Feigenbaum, 1987; Tweney, 1990).

Scientific discovery systems are also knowledge systems, and as the diversity and complexity of the tasks of a knowledge system increases the importance of knowledge organization becomes more evident. Such systems require different types of knowledge (e.g., facts, hypotheses, rules, events and processes) to be represented effectively.[3] In the short-term, real discoveries are more likely to be made by computational models that focus on one aspect of scientific discovery, such as planning empirical research, theoretical analysis, and theory formation and revision. On the other hand, if our aim is to build artificial research assistants, the problems of integrating different scientific research activities must be seriously considered.[4]

Acknowledgements

The author wishes to express his gratitude to John Fox and the other reviewers for their comments and suggestions.

References

Computational modelling of scientific discovery became known to a wider circle of reseachers in Europe through an excellent book titled *Scientific Discovery: Computational Exploration of the Creative Processes*, by Langley, Simon, Bradshaw and Zytkow (1987). The authors describe several computational models such as BACON, GLAUBER and STAHL in detail, and discuss the general strategies and methods involved in scientific discovery. This work was followed by another excellent collection titled *Computational Models of Scientific Discovery and Theory Formation* by Shrager and Langley (1990). This collection

[3] Elsewhere (see Kocabas, 1989), I proposed a methodology for organizing knowledge in systems that integrate various representations and are capable of conducting different learning tasks.
[4] Kocabas (1989) describes how such research activities can be carried out in an integrated way by a hierarchy of semi-autonomous research operators.

contains recent computational research in scientific discovery, and is interesting in its scope and depth as it tackles philosophical as well as methodological issues in the subject. The book is also interesting in its exposition of the speed of developments in the subject within a few years of the publication of the earlier work. By comparing the two books, readers will notice significant developments in such problems as structured knowledge representation, theory revision, and the integrated use of different methods of learning and discovery.

Another important work that can be expected to influence the progress in computational study of scientific discovery in an indirect way is *Building Large Knowledge Based Systems: Representation and Inference in the CYC Project* by Lenat and Guha (1989). The book explains the strategies and methods of representing a large amount of theoretical, technical and commonsense knowledge. More realistic discovery systems of the future will have to use a large amount of background knowledge, and Lenat and Guha's work can be helpful in dealing with the representational problems. Lenat's pioneering work on AM and EURISKO has been the source of motivation and ideas in the earlier work on machine discovery. However, the authors would have been better equipped in the CYC project if they were more knowledgeable in modern linguistic philosophy, as some of the issues in the categorization of knowledge had been resolved by philosophers such as Wittgenstein and Ryle.

Thagard's (1988) *Computational Philosophy of Science* is another first in the wider spectrum of the subjects related with the computational study of scientific discovery. The author discusses various issues in the philosophy of science from a computational perspective, describing a program (PI) that conducts conceptual and theoretical analysis. Thagard also provides an interesting criticism of "Evolutionary epistemology". I will not comment on the other publications listed below, as most of them are papers that can be read through relatively easily.

Buchanan, BG and Feigenbaum EA, 1978. "Dendral and Meta-Dentral: their application dimension" *Artificial Intelligence* **11** 5–24.

Buchanan, BG, Smith DH, White, WC, Gritter, R, Feigenbaum, EA, Lederberg, J and Djerassi, C, 1976. "Applications of artificial intelligence for chemical inference. XXII. Automatic rule formation in mass spectrometry by means of the meta-DENDRAL program" *Journal of the American Chemical Society* **96** (6168).

Davies, PCW, 1985. *The forces of Nature* (2nd ed), Cambridge University Press, Cambridge.

de Kleer, JR, 1986. "An assumption-based TMS" *Artificial Intelligence* **8** 127–162.

Dietterich, TG and Michalski, RS, 1983. "A comparative view of selected methods for learning from examples". In: Michalski, RS, Carbonell, JS and Mitchell, TM, eds., *Machine Learning: An artificial intelligence approach* Morgan Kaufmann, Los Altos, CA.

Forbus, KD, 1984. "Qualitative process theory" *Artificial Intelligence* **24** 85–168.

Friedland, P, 1979. "Knowledge-based experiment design in molecular genetics" *Proceedings Sixth International Joint Conference on Artificial Intelligence* 285–287.

Holmes, FL, 1980. "Hans Krebs and the discovery of the Ornithine Cycle" *Federation Proceedings* **39** 216–225.

Jones, R, 1986. "Generating predictions to aid scientific discovery process" *Proceedings Fifth National Conference on Artificial Intelligence* 513–516.

Karp, PD, 1990. "Hypothesis formation as design". In: Shrager, J and Langley, P, eds., *Computational models of scientific discovery and theory formation* Morgan Kaufmann, San Mateo, CA.

Kocabas, S, 1989. *Functional categorization of knowledge: Applications in modeling scientific research and discovery* PhD Thesis, Department of Electronic and Electrical Engineering, King's College London, University of London.

Kocabas, S, 1991 a. "Conflict resolution as discovery in particle physics" *Machine Learning* **6** 277–309.

Kocabas, S, 1991 b. "Homuncular learning and rule parallelism: An application to BACON" *Proceedings International Conference on Control* 950–954, IEE Conference Publications, London.

Kuhn, TS, 1970. *The Structure of Scientific Revolutions* University of Chicago Press, Chicago, 16.

Kulkarni, D, 1989. *The processes of scientific research: The strategy of experimentation* Doctoral Dissertation. Department of Computer Science. Carnegie Mellon University, Pittsburgh, PA.

Kulkarni, D and Simon, HA, 1988. "The processes of scientific discovery" *Cognitive Science* **12** 139–175.

Kulkarni, D and Simon HA, 1990. "The processes of scientific discovery" In: Shrager, J and Langley, P, eds., *Computational models of scientific discovery and theory formation* Morgan Kaufmann, San Mateo, CA.

Langley, P, 1978. "BACON1: A general discovery system" In: *Proceedings Second National Conference of the Canadian Society for Computational Studies.*

Langley, P, Simon, HA, Bradshaw, GL and Zytkow, JM, 1987. *Scientific discovery: Computational explorations of the creative processes* The MIT Press, Cambridge, MA.

Lenat, DB, 1979. "On automated scientific theory formation: a case study using the AM program" In: Hayes, J, Michie, D and Mikulich, LI, eds., *Machine Intelligence* **9** 251–283, Halstead, New York.

Lenat, DB, 1983 a. "The role of heuristics in learning by discovery: three case studies" In: Michalski, RS,

Carbonell, JG and Mitchell, TM, eds., *Machine Learning: An artificial intelligence approach* Morgan Kaufmann, Los Altos, CA.

Lenat, DB, 1983 b. "EURISKO: a program that learns new heuristics and domain concepts" *Artificial Intelligence* **21** (1–2) 61–98.

Lenat, DB, 1983 c. "The role of heuristics in learning by discovery: three case studies" In: Michalski, RS, Carbonell, JC and Mitchell, TM, eds., *Machine Learning: An artificial intelligence approach*. Morgan Kaufmann, Los Altos, CA.

Lenat, DB, Prakash, M and Shepherd, M, 1986. "CYC: using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks" *The AI Magazine* **7** (4) 65–85.

Lenat, DB and Feigenbaum, EA, 1987. "On the thresholds of knowledge" *Proceedings Tenth International Joint Conference on Artificial Intelligence*. 1173–1182.

Lenat, DB and Guha, RV 1989. *Building large knowledge based systems: Representation and inference in the CYC project* Addison Wesley, Reading, MA.

Michalski, RS, 1983. "A theory and methodology of inductive learning" In Michalski, RS, Carbonell, JG and Mitchell, TM, eds., *Machine learning: An artificial intelligence approach* Morgan Kaufmann, Los Altos, CA.

Michalski, RS, 1986. "Understanding the nature of learning: issues and research directions" In: Michalski, RS, Carbonell, JG and Mitchell, TM, eds., *Machine Learning* Morgan Kaufmann, Los Altos, CA.

Nilsson, NJ, 1965. *Learning Machines: Foundations of trainable pattern-classifying systems* McGraw-Hill, New York, NY.

Nordhausen, B and Langley, P, 1990. "An integrated approach to empirical discovery" In: Shrager, J and Langley, P, eds., *Computational models of scientific discovery and theory formation* Morgan Kaufmann, San Mateo, CA.

Nordhausen, B and Langley, P, 1987. "Towards an integrated discovery system" *Proceedings Tenth International Joint Conference on Artificial Intelligence* 198–200.

Omnes, R, 1970. *Introduction to particle physics* (Translated by G Barton) Wiley Interscience, London.

O'Rorke, P, Morris, S and Schulenburg, D, 1990. "Theory formation by abstraction" In: Shrager, J and Langley, P, eds., *Computational models of scientific discovery and theory formation* Morgan Kaufmann, San Mateo, CA.

Rajamoney, SA, 1990. "A computational approach to theory revision" In: Shrager, J and Langley, P, eds., *Computational models of scientific discovery and theory formation* Morgan Kaufmann, San Mateo, CA.

Ritchie, GD and Hanna, FK, 1984. "AM: a case study in AI methodology" *Artificial Intelligence* **23** 249–269.

Rose, D and Langley, P, 1986. "Chemical discovery as belief revision" *Machine Learning* **1** 423–452.

Rose, D and Langley, P, 1988. "A hill-climbing approach to machine discovery" *Proceedings Fifth International Conference on Machine Learning* 367–373. Morgan Kaufmann, Ann Arbor, MI.

Shrager, J and Langley, P, 1990. "Computational approaches to scientific discovery" In: Shrager, J and Langley, P, eds., *Computational models of scientific discovery and theory formation* Morgan Kaufmann, San Mateo, CA.

Simon, HA and Lea, G, 1974. "Problem solving and rule induction: A unified view" In: Gregg, L, ed., *Knowledge and cognition* Erlbaum, Hillsdale, NJ.

Stefik, M, 1978. "Inferring DNA structures from segmentation data" *Artificial Intelligence* **11** 85–114.

Stefik, M, 1981 a. "Planning with constraints (MOLGEN: Part 1)" *Artificial Intelligence* **2** 111–139.

Stefik, M, 1981 b. "Planning and meta-planning (MOLGEN: Part 2)" *Artificial Intelligence* **2** 141–169.

Thagard, P, 1988. *Computational philosophy of science* The MIT Press, Cambridge, MA.

Thagard, P, and Holyoak, K, 1985. "Discovering the wave theory of sound: Inductive inference in the context of problem solving" *Proceedings Ninth International Joint Conference on Artificial Intelligence* 610–612.

Thagard, P and Nowak, G, 1990. "The conceptual structure of the geological revolution" In: Shrager, J and Langley, P, eds., *Computational models of scientific discovery and theory formation* Morgan Kaufmann, San Mateo, CA.

Tweney, RD, 1990. "Five questions for computationalists" In: Shrager, J and Langley, P, eds., *Computational models of scientific discovery and theory formation* Morgan Kaufmann, San Mateo, CA.

Zytkow, JM and Simon, HA, 1986. "A theory of historical discovery: the construction of componential models" *Machine Learning* **1** 107–137.

Zytkow, JM, 1990. "Deriving laws through analysis of processes and equations" In: Shrager, J and Langley, P, eds., *Computational models of scientific discovery and theory formation* Morgan Kaufmann, San Mateo, CA.