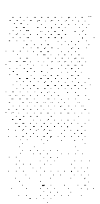
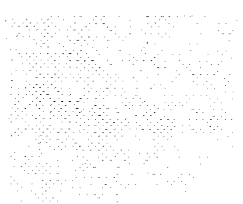
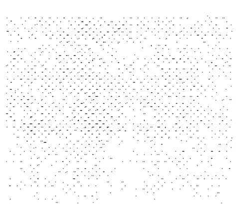
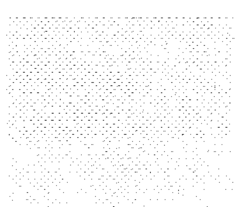
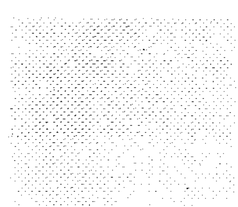
Signs, Search and Communication

Semiotic Aspects of Artificial Intelligence

Edited by
René J. Jorna
Barend van Heusden
Roland Posner

Offprint

Walter de Gruyter · Berlin · New York
1993

Expert systems and the abductive circle

G. LUGER & C. STERN (Albuquerque)

0. Abstract

The current generation of expert systems is becoming more and more successful at functioning in a collaborative role with human users in abductive problem solving. We first show how these systems implement a symbol processing architecture which allows for a cyclical pattern of description, explanation, and action. We next analyze the structure of abductive problem solving and evaluate the ability of current expert systems to engage in a semiotic interpretation of evidence. We show where epistemological understanding has not kept pace with expert system problem solving practice. Finally, we argue that a new epistemology, based on a clearer understanding of abductive problem solving, could provide a basis for integrating many of the tools now emerging from areas of A.I. research.

1. Introduction

Expert systems problem solving was one of the first A.I. technologies to migrate on a large scale from the research environment to a commercial setting. An important part of the widespread use and success of expert systems has been their adaptability to many problem solving contexts. This adaptability has involved the integration of a variety of knowledge representation and reasoning techniques.

In the section 1.1 we trace the roots of A.I. programming techniques in structuralist models of problem solving. We examine exploratory programming, proof trees and rules stacks, and show how these tools have been used to describe and explain the problem solving process. We then show how much of the versatility of expert systems derives from the robustness of these original techniques. In section 1.2 we point out many of the current limitations of expert systems, including brittleness, and lack of meta-level integration, and the inability to model common sense.

In section 2 we analyze abductive problem solving, describing an architecture which includes interwoven elements of description, explanation and intervention. We give an example from our own expert system development work of abductive problem solving in the area of discrete semiconductor failure analysis.

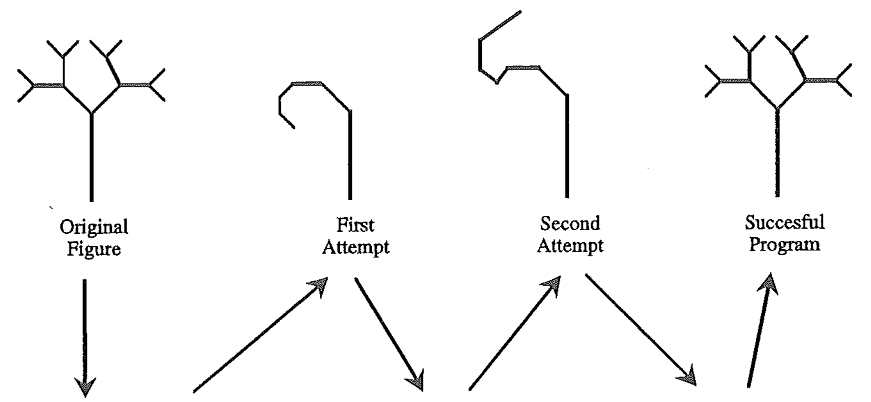
Finally, in section 3, we propose an epistemological foundation for current expert systems practice that offers a possibility of extending their semiotic sophistication.

1.1 Expert systems: a "use driven" technology

Seymour Papert in the mid-1960s described for the computing and education communities what has come to be referred to as "exploratory programming". Papert exemplified this approach with his LOGO language coupled with the use of "turtle graphics" and an interactive programming environment. Papert first proposed this methodology as part of a theory of education where a program actively represented a student's reasoning process in a particular situation and the computer's interpretation of the student's code allowed the student to comprehend problems, called "bugs," in his or her thought process (Papert, 1980). Figure 1 shows a student's code/thought of a tree with the computer's interpretation demonstrating "bugs" in the thought process. Progressive refinement of the tree code, first adding a recursive termination condition, and then creating a "procedure invariance" so that the turtle ends each recursive call having the same orientation with which it started, finally produces the intended result.

Papert's ideas are built on the Piagetian (structuralist) paradigm in psychology, where humans learn new relationships in their world through a process of accommodation to discrepancies in the structures of their current understanding. In this "genetic epistemology" new invariances constantly update and replace current limited (buggy) understandings of situations. Papert translated these ideas into a computational environment and they subsequently became an important aspect of the artificial intelligence and expert systems programming methodology.

Expert systems designers use exploratory programming under several guises, but most importantly in exploring the problem space and in developing the early prototype. We explore the problem space of an application to determine the salient features for a program design. These features might include the choice of data or goal driven inferencing, of breadth, depth of best first search strategies, or representation through decision trees, rule systems, objects or some hybrid combination. Throughout this process we seek to align the computer's interpretive process with that of the human expert interpreting the phenomena of the problem situation.




```
TO TREE :SIZE
FD :SIZE
LT 20
TREE :SIZE/2
RT 40
TREE :SIZE/2
END
```

```
TO TREE :SIZE
IF :SIZE<2 [STOP]
FD :SIZE
LT 20
TREE :SIZE/2
RT 40
TREE :SIZE/2
END
```

```
TO TREE :SIZE
IF :SIZE<2 [STOP]
FD :SIZE
LT 20
TREE :SIZE/2
LT 20
BK :SIZE
END
```

Figure 1: The progressive refinement method used to develop the computer code for drawing a tree

Once the base representation and search strategies are selected the expert system designer begins to fashion the rules or knowledge structures of the particular application. In developing the early prototype the A.I. programmer attempts to fashion a knowledge base reflective of the expert's interpretation of the application. In this situation it is perfectly acceptable to say, "Even though I don't know the full details of what I am doing, let me explore with a few ideas..." Through this progressive refinement model, the computer's interpretation, directly reflecting the programmer's evolving understanding of the domain, gradually comes to capture the understanding and interpretative skills of the human expert working in that application area.

Exploratory programming has come to be one of the designer's most important tools. John McDermott, in reflecting on his preliminary work and successes in building the computer configuration program, XCON, at Digital (McDermott 1981; McDermott & Bachant, 1984) insists that the exploratory programming methodology made his program possible, whereas two previous attempts by Digital, not using a very high level language and the exploratory methodology, had failed at the task.

We include as Figure 2 a flow chart describing the exploratory programming model. There are two important features: first, try an idea and see if it works, in particular "watch where it fails", and revise (the code representing) your idea accordingly.

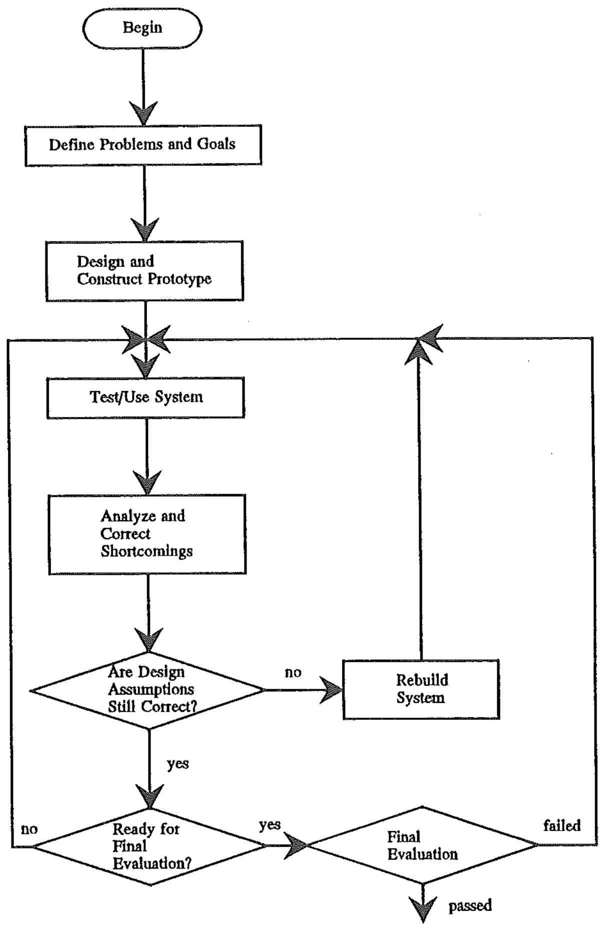


Figure 2: The explanatory development cycle for rule based expert systems

The second feature is, once the designer has worked through the thought process until the application is understood, be ready to scrap the rule or object design that resulted from the exploratory mode, and rewrite it from the beginning to reflect directly the new understanding of the problem situation. When McDermott did this with an early version of the XCON program he found that he could build a program of identical competencies with one third fewer rules, from 750 down to 500 (McDermott, 1981).

An important A.I. tool to assist in the exploratory approach is a powerful user interface design. With the object-oriented interface first seen in Interlisp and the Smalltalk world at Xerox PARC, and later reflected in the Flavors environment at MIT, active tracing and debugging tools have became a part of the expert system designer's tool kit. Besides offering an important assist in removing syntactical miscues, the interface allows the designer to oversee the computer's interpretive process. With the assistance of a sophisticated interface, the program designer is able to view failure within the context of the interpretive process.

The exploratory programming methodology, combined with the rule or object centered data structure, makes it possible for the program designer to view an appropriately grained analysis of miscues. This allows the programmer to see misconceptions in the context where they occur. A proof tree or rule stack gives the exact location in the computer's interpretative process of the programmer's misconception. This means, of course, that the program designer is dealing not just with some abstract mistake (array variable out of bounds, stack underflow) but a conceptual mistake in the context of an interpretation (this disk drive requires two controllers and there are no such devices available).

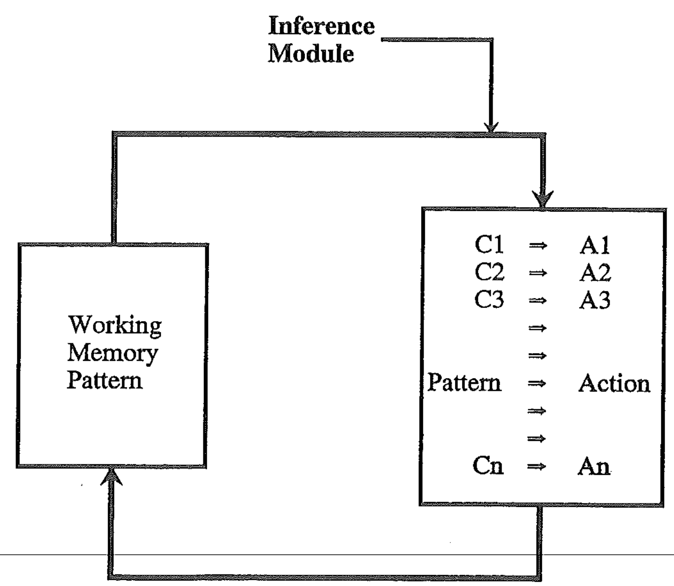


Figure 3: The production system architecture. The inference module takes patterns from working memory to the rule set and returns revised patterns

To make this notion of "interpretive context" for rule debugging clear, we offer a simple example. Using a expert system controller such as the traditional production system, rule based inferencing exactly reflects progress through an and/or graph, and that graph gives the context for each rule as it is interpreted. Figure 3 presents a diagram of the production system model of computing with its traditional division of logic (the encoded rules of expertise) from inference engine. The inference module goes to the rule base with the context of the information in working memory to match a production rule and generate another step in the search process.

The rule stack, as part of working memory, reflects the progress through the graph of possible solutions. With each successful rule invocation, that rule is placed on the stack. Thus, if the user is asked a question (does the probe light register positive?) the program can present the rule as the justification as to why the question was asked (because if the probe light registers positive and ... then the fault is in ...). If and when a rule later fails to be part of a successful solution path, it is popped off the rule stack. Thus the stack reflects both the chaining of rules together and the context of any particular interpretation. A reflection of the rule stack as progress through a graph is given as figure 4.

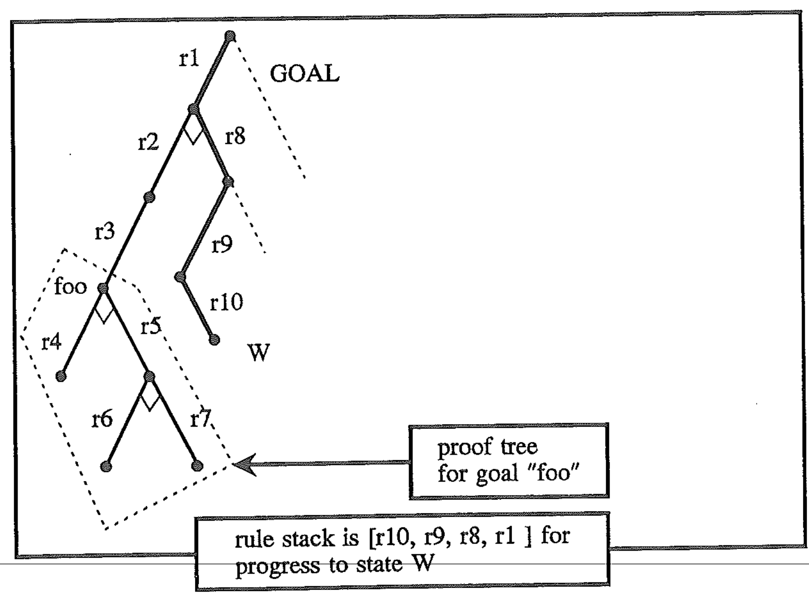


Figure 4: Proof tree that reflects reasoning

The result of any solution, or even partial solution in the problem is reflected in a proof tree. A proof tree is that portion of the graph, along with the path through its states, that captures the completed reasoning process in the problem or subproblem. When the program designer and later the user asks the system "How did you come up with this particular result?," the program responds with the proof tree that reflects its reasoning in that situation. This situation is captured in Figure 4, where the user has asked the program how it knows "foo" is true. The encircled portion of the graph is the system's response.

Two important points can be made from the analysis of the proof tree and rule stack concepts: First, in exploratory programming mode, the designer or programmer has an interpretive context for understanding her misconceptions or bugs. Her mistakes are made in a context, and interpreting and correcting them is a conceptual accommodation (in the full genetic epistemological sense) to the world as it is. The second point is that the eventual expert system user is also always problem solving in a context. Accompanied by a rule stack with solutions reflected in proof trees, inference rules and explanations are fundamentally linked to a focus or context, that is, they are interpretations.

The explanation/justification powers of the expert system methodology are a hallmark of this genre of programs: if the justification of decisions in the search process is not available, the program is not an expert system. As we see in subsequent sections explanations are also an essential part of human communication, and even more mature justifications will be part of our description of the abductive circle: requirements for a symbol system to achieve semiosis.

Current research in expert systems is attempting to address difficult problems, such as the design of intelligent editors to help, by automation of the process, the human domain expert interact with the expert system program in debugging the knowledge base. The Teiresias work at Stanford (Davis, 1982; Davis & Lenat, 1982) shows us an example. A "metatheory" of the MYCIN rule base was built for the domain expert to reason about the objects (rules and facts) in the MYCIN knowledge base. Using "schemas" the Stanford researchers built explicit models of classes of rules. These schemas contained information on how to create new instances of a class of objects, how information about a set of rules is to be obtained, how certain infections can enter the system, and so on. In the dialogue below, when the doctor attempts to enter a new rule in the knowledge base, Teiresias is able to "understand" it, attempt to enter it into a class of already existing rules, and to determine why the rule is not appropriate.

T: Is there a rule missing?
D: Yes

T: Would you like to give me such a rule now?
D: Yes.

If 1. The patient's infection is primary-bacteremia
2. The site is one of the sterile sights
Then There is suggestive evidence (.8) that the category is enterobacteriacae.

(Teiresias parses this rule into its internal representation and then presents it again so the expert can check it. After the expert agrees, Teiresias attempts to integrate this new rule into the knowledge base....)

T: I hate to criticize, Dr Davis, but did you know that most rules about what the category of an organism might be that mention the site of a culture, and the infection also mention the portal of entry of the organism? Shall I try to write a clause to account for this?
D: Yes
T: How about:
The portal of entry is gastrointestinal. OK?
D: Yes

The proof tree and rule stack are also used by Teiresias, as one would expect, in the debugging process. Teiresias asks the user if a particular rule stack or proof tree justification are appropriate in a situation. Teiresias is in effect asking the human doctors if their interpretation is the same as MYCIN's interpretation of a particular medical situation! Teiresias also provides a "meta-meta" level where new schemas, for new classes of rules can be created.

In spite of this growth and widespread adoption of the expert systems methodology, there has been no unifying computational epistemology that has kept pace with these evolutionary developments. The theoretical understanding of expert system problem solving has remained rooted in the denotational semantics of Tarski and the deductive model of human problem solving.

Even the so called "uncertain" reasoning systems, using techniques such as the Stanford Certainty Factor Algebra (Buchanan & Shortliffe, 1984) and Zadeh's fuzzy set theory (Zadeh, 1983; Luger & Stubblefield, 1989), are seen as weak deductive methods rather than as genuine instances of an inference process. Many expert systems technologists, with their rationalist origins, seem comfortable only with deductive inference. Sound inference schemes may make good mathematical theorem provers but they make little psychological sense. We believe that these and other problem solving processes which expert systems use are by and large abductive in nature. Furthermore, we feel that the epistemological foundations of expert problem solving requires a more sophisticated account of the functioning of symbol systems and the role of interpretation.

In this context we will define the abductive circle. Roughly, this term refers to the linkage of description, explanation, and action within the context of abductive problem solving. We believe that the need for expert systems to capture this dimension of problem solving in humans has motivated the incorporation of features that support abductive problem solving. We will propose a theoretical account of abduction which we believe will provide direction for the future evolution of expert system technology. Before we do this we will briefly mention several of the current limitations of the expert or knowledge based technology.

1.2 Current limitations of the expert systems technology

Despite their successes, expert systems are still subject to a number of fundamental limitations. We see these limitations as a reflection of the semiotic poverty of our current knowledge representation and reasoning techniques. We enumerate several of these limitations before continuing with our definition of abductive circularity.

Brittleness: This correlates with simplistic sharp-edged concept modeling. Brittleness comes from two aspects of the current technology. First, a Tarskian semantics that maps elements of a symbol system onto items "in the world." We feel this simplistic mathematical approach to a semantic model is fundamentally flawed, as we discuss in subsequent sections. Second, the matching algorithms in expert systems, so crucial for chaining rules as well as matching rules to descriptions of the world, are performed by simplistic, context independent techniques such as unification (Luger & Stubblefield, 1989). Matching is never within a context that could support near (syntactic) misses, alternative (equivalent) descriptions, or any guidance based on deeper (semantic) understanding.

Shallow versus deep reasoning: The "shallowness" of a problem specification is sometimes related to surface or descriptive, rather than deep, explanatory, or semantic representations. In MYCIN, for example, medical situations are often described by temperature levels and presence of nausea and headaches, rather than explained through a theory of bacterial infecting agents. The failure to produce a solution with the shallow model is sometimes responded to by model-based reasoning (Iwasaki, 1989). Transfer between different levels of reasoning remains an unsolved problem due to the lack of semantic cohesion between the representational levels (Skinner & Luger, 1991).

Weak explanation facilities: This is related to the previous point. The best expert systems currently produce explanations which are built from literal statements of rules from the rule stack or rebuilt into proof trees. Explanation in humans is a much richer process, often including deeper clues to the "intervention process" and the eventual problem solution. For example, human explanations often include contexts for switching between shallow and deep reasoning models (Skinner & Luger, 1991).

Lack of (semantic) metalevel integration: Rule based expert systems, or even rule sets attached to object based specifications are little more than a loosely coupled collection of information. We refer to this as the "wood-pile" model of skill representation: when you want to encode a new skill just write it in an if.. then.. format and throw it into the pile of other rules. Rules can be modular and "additive" to the point of absurdity. What is desired is (at the minimum) more metalevel coherence. The exception to this approach, and important data points in meta level modeling are the Metadendral and Teiresias research (Lindsay et al., 1980; Buchanan, 1984).

Common sense: Common sense in human problem solvers functions in background to support flexibility of expertise. We don't know how to model this background knowledge very well. This skill is related to our embodiment in the world and our repository of basic motor and coping skills (Winograd & Flores, 1986). We discuss these ideas further in subsequent sections.

Difficulty in paradigm selection for expert reasoning systems: At the present time there does not exist a mechanism under which we can decide which problems or classes of problems will be amenable to the expert systems approach. In many cases the way to deal with this decision is to engage in exploratory programming, a semiotic exploration of the problem solving space.

Lack of learning potential: The present generation of knowledge based systems lacks even rudimentary potential to automatically integrate new information, even that from successful runs, into its knowledge base. Current research in model and case based reasoning offer hope in this regard.

The account of abduction offered in this paper we believe will show both the sources of these limitations and a means for addressing and overcoming them.

2. Expert problem solving and abduction

2.1 Introduction

Expert systems manipulate symbolic representations. Their computations are symbolic in two senses. First, they manipulate non-numeric symbols representing qualitative properties and relationships. Second, it is intended that these symbols have the same internal or computational semantics as that attributed to them by the human user.

In order to encode the knowledge of an expert, the expert system designer must have a clear view of the semantics of the symbols which the expert uses. In this section we shall argue that such a view requires an understanding of the functioning of symbol systems within a fundamentally abductive problem solving process.

Symbol systems serve as instruments for the effective interaction of experts and their problem-solving domain. They organize and codify the domain phenomena as well as the expert's understanding and response to those phenomena. The semantic structure of the expert's symbol system cannot be divorced from the problem solving knowledge to which it gives form. The three phases of problem solving, description, explanation, and intervention, are inextricably linked together as three aspects of a single problem solving practice. What weaves them together is an abductive structure embodied in the semantic linkages of the expert's symbol system.

We now flesh out these observations by showing that the semantics of the expert's language is defined by its relationship to description, explanation, and action, and that this semantics carries the linkages between these three phases of the problem solving process (see figure 5). We also observe the abductive character of this problem solving in the dynamic alternation of these phases.

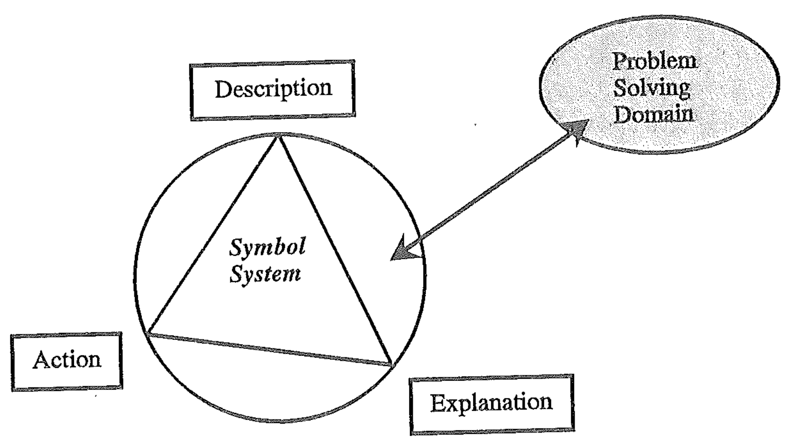


Figure 5: The abductive circle

Description:

Problem solving begins with a description of the problematic situation. This description is more than a list of facts innocently gathered and assembled from the problem solving environment. The effective selection of facts and the ordering of these facts into a coherent description presupposes 1) an ability to recognize in the situation features and constellations of potential significance and 2) fluency in a set of domain specific descriptive forms. Novices can rarely recognize and isolate the salient features of a situation, and their descriptions are often full of omissions and irrelevancies. This is of course why doctors and auto mechanics refuse to make diagnoses on the basis of a layman's description. Domain experts stand indeed further above their less skilful colleagues in their ability to construct descriptions which are at the same time both more concise and more revealing.

Problem solving generally begins with an initial description of the problem solving situation. This description is then developed and elaborated in conjunction with a process of inquiry. New information is acquired through measurement, testing, interviewing, etc., information which is then reintegrated into the situation description.

The acquisition and integration of new information has associated costs in time and resources. This process thus requires focus and direction from a goal or set of goals in order for the effort to efficiently converge. These goals are formed by the expert's response to the current state of the description, its completeness or incompleteness, its coherent or anomalous character, the categorizations and explanations which it suggests. At various stages of the inquiry, the expert will adopt an hypothesis or set of hypotheses. These then direct the course of inquiry until they are refuted or replaced by more probable hypotheses. This means that the elaboration of the description is an ongoing, abductive process organized and informed by the expert's domain knowledge.

Description and explanation do not represent distinct and separate processes but rather locations in a continuum. Description evolves, in the process of inquiry and elaboration, in the direction of explanation. At each stage in the process the description projects, through its structure and content, a certain presence or absence of intelligible form. The description may initially be sketchy and incomplete, too inchoate to allow for any explanation. Later it may contain aspects which are coherent and which suggest a spectrum of possible explanations, and others which are anomalous and which drive inquiry further. Finally, if the inquiry is successful and the problem situation matches closely one of prototypical or well understood cases, then the description may be so lucid and self-explanatory as to require only the most perfunctory explanation. The progress of the problem solving process thus involves an adaptive interaction between descriptive and explanatory forms, an interaction mediated in part by the semantic relations implicit in the expert's symbol system.

Explanation:

Explanation of a situation is relative to description of the situation: how we specify *what* is to be explained partly determines *how* it is to be explained. The relationship is, however, bidirectional. Explanation may locate key features of the situation which determine others; this may in turn lead us to change the organization and structure of our description.

Classical theories of explanation have tended to focus on law and causality. Hempel (1965), for example, elaborates a view of scientific explanation as a process of connecting the particular situation to the operation of general or universal laws of nature. The general form of explanation is thus that of subsuming the particular under the universal. (There is a rough correspondence between this and the proof tree model of explanation in current expert systems: The rules involved in the proof correspond to the "universal laws".)

Currently work is underway in A.I. to develop a formal model of abductive explanation. One account is that of Levesque (1989) who proposes a definition of abductive explanation in terms of a (mathematical) relation called EXPLAIN. The sense of this relation is given by the axiom:

$$\mathrm{EXPLAIN}_e((P \rightarrow Q), Q, P)$$

which is interpreted as follows: If, in a given epistemic state *e*, P together with the background beliefs of that state entails Q, then P is one explanation for Q relative to *e*. We have shown (Stern and Luger, 1991) that entailment-based accounts of explanation fail to capture the semantic characteristics of *good* explanations, e.g., edification (non-triviality), clarity and coherence.

To this we add that the characterization of explanation must be domain- and context-specific. Expert explanations take many different forms (not merely subsumption of the particular under a universal rule). The defining feature that all these differing forms have in common is that they serve a purpose in the expert's problem solving process, providing direction for further inquiry or problem solving intervention.

Abduction, hypothesis, and intervention:

Problem solving under limitations of time and/or resources requires a special form of reasoning and decision making. The expert must navigate in conditions of uncertainty, using a method of informed hypothesis to find an expeditious path to a solution. While uncertainty is an unavoidable concomitant of the journey, it must be minimized by the time the destination is reached.

The formulation of an hypothesis serves either as a means to focus the course of inquiry or as a prelude to action. In both cases, the hypothesis stands between the problem description, on the one hand, and a decision or commitment on the other. The hypothesis functions as an explanation of currently known facts and at the same time as the ground of a commitment either i) to the verifiability of certain new facts or ii) to the success of a certain form of intervention. In both cases, the hypothesis represents the abductive gathering of known and unknown, of past and future.

Hypothesis formation relies both on a body of theory and a body of experience. While the theory provides the descriptive terms and linkages which comprise the fabric of the explanation, experience provides a set of successfully handled or well-explained cases which schematize the application of the theory. Often the formulation will begin with the postulation of a relationship between the current instance and some previous instance(s), a set of correspondences and differences which locate the current instance in a space of paradigmatic cases.

The case-based hypothesis is then elaborated through a mapping process, where the domain theory together with these correspondences are used to extract testable consequences or a tentative course of action. It is generally here that the expert's skill is most sharply distinguished from that of the newly schooled novice. While the novice may possess the same abstract level of theoretical understanding, the expert's richer background of interpreted experience allows her to identify suggestive patterns in the domain phenomena and to formulate hypotheses which focus problem solving more quickly and effectively.

The expert's problem solving is organized to serve two purposes: to provide effective problem resolution and also to provide significant feedback about the problem solving situation. If the intervention is unsuccessful, the results of the intervention should provide as much information as possible about what didn't work and why. The interpreted results should be able to inform the direction of continued efforts.

Even if the intervention is successful, the results of the intervention must still be evaluated. The expert must *know* that it is successful and have a clear understanding of why it worked. The expert must have confidence in the solution and a well defined set of expectations about when it will work again in the future. The reason for this is that the expert's problem solving is not, in general a "single shot" process; it is part of an ongoing evolving practice. To achieve the capacity for adaptability and growth, the expert's interaction with the environment must be organized from the outset dialogically. This involves a semiotic structure in which the outcome of the interaction is itself a message, a carrier of useful information which can be folded back into the expert's problem solving representations.

2.2 Abductive problem solving: an example

In this section we present an example of expert problem solving derived from one of our current projects. We are designing a "discrete semiconductor failure analysis expert system" for Sandia National Laboratories. "Discrete semiconductor" here refers to transistors and diodes as opposed to integrated circuits.

Failure analysis, in this case, is generally performed on a device with which the expert is already familiar. Either he or someone in his work team was involved in drafting a procurement specification for the device and in evaluating original manufacturer's samples. Based on this initial understanding of the device, the expert is responsible for monitoring the device through its life cycle. This means assisting in the handling of problems in the manufacturing process, in incoming-receiving inspection, circuit assembly, and finally, in the field.

Detecting the source of failures requires demanding detective work. There are several different potential problem sources. These include:

i) Problems in the manufacturing process, such as volatile or particle contamination, improper bonding or welding, improper or inconsistent silicon doping, or a cracked silicon chip.
ii) Device level design problems, such as adjacent materials which are chemically reactive or bonded materials with different coefficients of expansion.
iii) Screening problems[1], such as a screen that is not tight enough or does not cover the right characteristics.
iv) Handling problems, such as damage due to faulty test equipment, rough handling during installation, or electrostatic damage.
v) Circuit problems, such as the circuit damages the device through overexercise or improper temperature control.
vi) Circuit level design problems: the device is the wrong part for the job.

A typical problem solving process proceeds through the following series of stages:

i) Evaluation of failure background, including circumstances of the failure and past history of the device.
ii) Failure verification by establishing that the device indeed failed in the way that was reported.
iii) External physical analysis: a precise description of the device's external physical and electrical characteristics.
vi) Internal physical analysis. This usually involves opening up the package and examining the device with optical and electron microscopes or performing chemical analyses; it is generally a destructive process.
v) Causal analysis that determines the source of defects or damage at the physical level.
vi) Developing a fix or cure for the problem.

The first phase in the expert's problem solving is to assemble and assess background information. Where did the device fail? After what period of testing or use? How many devices failed? Are the circumstances of failure similar to some previous event for the same device? For a similar device? Do the circumstances of failure suggest a pattern, e.g., the manufacturer has a history of contamination problems, or many low power devices have been failing after testing on a certain piece of test equipment?

By the time failure analysis reaches the stage of electrical testing, it has already become, to a great extent, data-driven. Because this testing is time consuming, and more importantly, because it can cause further damage to the device, thus destroying the evidence, the selection and ordering of electrical tests requires a good deal of skill. The choice of test parameters as well as voltage and current levels is determined, for each test, by the results of previous tests and the expert's reading of their contextual significance. The device's electrical characteristics are interpreted as indicators of damage or defect at the level of physical structure. The tests are orchestrated so as to narrow or constrain the range of hypothesis about physical structure while at the same time preserving the possibility of pursuing alternative hypotheses should the current ones fail.

Not until progress has been made in identifying a strongly suggestive pattern of evidence at the external electrical and behavioural level is physical analysis undertaken. This is because i) physical analysis is irreversibly destructive, and ii) because the required physical evidence is commonly located at the microscopic level and hence cannot be found without first knowing where to look.

The final step in failure analysis is the identification of the cause and offering a cure. These often proceed hand-in-hand. Conceptually, of course, the identification of cause should determine the selection of cure, but in actual practice it is often the case that a tentative form of problem intervention is undertaken partly to verify the correctness of the causal analysis.

For example, if a small number of failures of a certain component begin to appear after some period of time, it might be hypothesized that a few marginal devices which barely passed an original screen eventually degraded through the rigors of use. The cure, on this analysis, would be to tighten the requirements of the screening procedure so that marginal devices are screened out. This "cure" has the virtue of simplicity and relative low cost. Were this kind of "cure" to fail, moreover, the failure itself would be revealing. It would indicate that the failures are "latent", meaning that apparently good devices experience significant degradation over time. Since "latent failures" have a restricted range of possible causes, this information would be quite useful.

2.3 Reflections and analysis

One of the striking features of the analytical process described above is the progression of stages (figure 6).

In the initial failure report what is described are gross physical or electrical behaviours. Extensive testing then gives a more and more detailed and precise account of the device's external physical and electrical characteristics. This characterization not only illuminates the earlier failure event but also serves as a road map to the hidden landscape of physical structure. Internal physical analysis then encounters that landscape firsthand, discovering the structural causes of these physical and electrical characteristics.

Levels of Description	
Initial failure report	Rough description of failure event, e.g. intermittent base emitter open, degraded BV
External physical and electrical analysis	Precise characterization of electrical charateristics, package intergrity, residual gas analysis
Internal physical analysis	Delid package and examine under electron microscope; perform chemical analysis
Causal failure analysis	Use physical indications to decipher environmental causation

Figure 6: Progression of stages as a feature of the analytical process

Finally, causal analysis uses indications from the level of physical morphology to track down the sources of these physical defects, whether these factors be traceable to design flaws, manufacturing problems, mishandling, or other environmental influence.

The structure of inquiry organizing this progression is thus a semiotic spiral (Figure 7) in which each level of description illuminates its predecessor while at the same time anticipating and pointing to its own explanation at the next higher level.

The implicit reference of level to level is registered in many of the descriptive terms which are used. A term like "overstress", for example, refers both to an excessive voltage or current applied to a device and also to the pattern of distress that such an excessive voltage or current can produce.

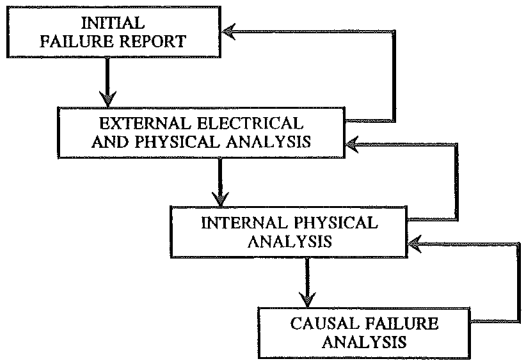


Figure 7: The abductive spiral

This pattern can be a recognizable physical indicator of a past event. Thus the expert talks about a "signs of overstress" or "overstress damage" visible under a microscope. "Volatile leakage" is another example of a physical indicator. "Volatile leakage" refers to a leakage characteristic of reverse breakdown voltage which improves with elevated temperature.

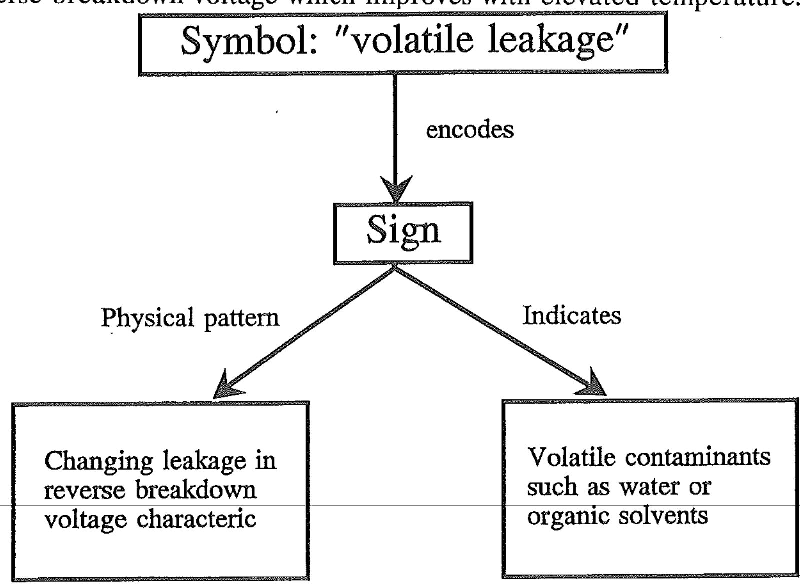


Figure 8: Relation between symbol, sign and physical reality

"Volatile leakage" provides clear indication of the presence of volatile contaminants such as water or organic solvents (Figure 8).

Terms such "soft knee", "intermittent open", "walking breakdown", etc., also encode the semiotic relationship between a physical pattern (generally on a laboratory instrument) and the causal factors to which that pattern points. It is in this sense that the expert's symbol system embeds his problem solving knowledge. Although we do not have the time to expand the observation here, we further note that the expert's instrumentation has been designed to perspicuously display those patterns deemed indicative or significant. In this sense, the expert's explicit linguistic representations are themselves comprised within a larger system of signs which organizes the expert's problem solving environment.

3. Areas of research and growth

As a result of the entry of expert system technology into the commercial marketplace, it has undergone an evolution that has moved it well beyond the goals of its original designers. Its epistemology has also outgrown its rationalist foundations (Luger and Stern, 1990)

In spite of its successes, there remain some fundamental flaws and limitations in expert system technology. We see this in its brittleness and its lack of ability to provide deeper justification than that of proof trees and rule stacks. We also note the absence of a rule/model symbiosis that would allow the system to employ either technology when appropriate. Finally, it is still well beyond the ability of current expert systems to improve performance through accumulated experience. We would like to see some limited learning ability in the next generation of expert systems, at the very least the integration of successful cases into its knowledge and practice.

Early efforts in A.I. were biased towards deductive inference. Limitations of deduction in areas such as diagnosis and theory formation have led to increased interest in abductive inference. Our description of the "abductive circle" provides the basis for a more realistic understanding of abductive problem-solving and the symbol systems which support it. In our ongoing work (Stern and Luger, 1991) we are developing a sign-based account of abductive explanation which offers significant advantages over the current entailment-based account.

The research challenges ahead necessitate a re-evaluation of the epistemological foundations of expert system problem solving. The epistemologies prevalent in A.I., based on formal logic and Tarskian semantics, are characterized by a self-imposed poverty which limits their ability to grasp the implications of current practice. Our analysis of abductive problem solving comes closer to a structure for understanding current expert system tools and practices. Our analysis, coupled with some of the techniques now emerging from A.I. research (Luger and Stubblefield, 1992), offer the promise for understanding and creating more semiotically mature computational systems.

Note

1. It is not cost effective to eliminate variability in the semiconductor manufacturing process; it is more cost effective to screen out the small percentage of a manufacturing lot which is lower quality.

References

Buchanan, B., Shortliffe, E. (eds.) (1984). *Rule Based Expert Systems: The MYCIN Experiments of The Stanford Heuristic Programming Project*. Reading, MA: Addison Wesley.

Davis, R. (1982). Applications of meta level knowledge to the construction, maintenance and use of large knowledge bases. In: Davis & Lenat 1982.

Davis, R., Lenat, D. (1982). *Knowledge Based Systems in Artificial Intelligence*. New York: McGraw Hill.

Eco, U. (1976). *A Theory of Semiotics*. Bloomington: Indiana University Press.

Hempel, C.G. (1965). *Aspects of Scientific Exploration and Other Essays in the Philosophy of Science*. N.Y.: Free Press.

Iwasaki, Y. (1989). Qualitative physics. In: A. Barr, P. Cohen. and E. Feigenbaum (eds.), *Handbook of Artificial Intelligence*, vol IV, Palo Alto: Addison Wesley.

Levesque, H. (1989). A knowledge level account of abduction. In: *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, San Mateo, Calif.: Morgan Kaufmann.

Lindsay, R.K., Buchanan, B. G., Feigenbaum, E. A., and Lederberg, J. (1980). *Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project*. New York: McGraw-Hill.

Luger, G.F., Stern, C. (1990). *The Role of Interpretation in the Representational Methodology* (Tech Report, Computer Science Department, University of New Mexico, Albuquerque New Mexico).

Luger, G.F., Stubblefield, W.A. (1989; 1992 sec. ed.). *Artificial Intelligence and the Design of Expert Systems*. Palo Alto: Benjamin Cummings.

McDermott, J. (1981). R1: The formative years, *AI Magazine*.

McDermott, J., Bechant, J. (1984). R1 revisited, four years in the trenches, *AI Magazine,* 5:3.

Papert, S. (1980). *Mindstorms*. New York: Basic Books.

Peirce, C.S. (1955). *The Philosophical Writings of Peirce*. New York: Dover Publications.

Skinner, J., Luger, G. (1991). A synergistic approach to reasoning for autonomous satellites. Presented at The Conference of Machine Intelligency for Aerospace Electronic Systems, Lisbon, 1991.

Stern, C., Luger, G. (1991). Towards a Sign-based Theory of Abduction. (submitted to the 1991 Conference on Robot Epistemology, copies available from authors).

Tarski, A. (1944). The Semantic Conception of Truth, *Philosophy and Phenomenological Research*, Nr. 4, 341-375.

Winograd, T., Flores, F. (1986). *Understanding Computers and Cognition.* Norwood NJ: Ablex.

Zadeh, L. (1983). Commonsense knowledge representation based of fuzzy logic, *Computer*, Nr. 16, 61-65.

Contents

III Communication with Expert Systems