59 | 1–2 | 02

Brain, Behavior and Evolution

The Evolution of Intelligence: Brain, Behavioral and Computational Approaches

21st Annual Krost Symposium,
Seguin, Tex., March 22–23, 2001

Guest Editor
Scott Bailey, Seguin, Tex.

S. Karger
Medical and Scientific
Publishers
Basel • Freiburg
Paris • London
New York • New Delhi
Bangkok • Singapore
Tokyo • Sydney

KARGER

Brain Behav Evol 2002;59:87–100

Problem Solving as Model Refinement: Towards a Constructivist Epistemology

George F. Luger Joseph Lewis Carl Stern

Department of Computer Science, University of New Mexico, Albuquerque, N. Mex., USA

Key Words
Artificial intelligence · Constructivist epistemology · Human complex problem solving · Model refinement · Diagnostic reasoning · Intelligent control

Abstract
In recent years the Artificial Intelligence research group at the University of New Mexico have considered several areas of problem solving in interesting and complex domains. These areas have ranged from the low level explorations of a robot tasked to explore, map, and use a new environment to the development of very sophisticated control algorithms for the optimal use of particle beam accelerators. Although the results of our research have been reflected in computer-based problem solvers, such as the robot discovering and mapping out its world, these computational tasks are in many ways similar to expert human performance in similar tasks. In fact, in many important ways our computer-based approach mimics human expert performance in such domains. This paper describes three of these task domains as well as the software algorithms that have achieved competent performances therein. We conclude this paper with some comments on how software models of a domain can elucidate aspects of intellectual performance within that context. Furthermore, we demonstrate how exploratory problem solving along with model refinement algorithms can support a constructivist epistemology.

Introduction

We often find ourselves struggling for a definition of, or explanation for, expert problem solving performance, of expertise practiced in context. We may even find ourselves reluctant to use terms such as 'intelligence' or 'expert problem solving' to describe the actions of dolphins, non-human primates, colonies of bees, or even simple human performance. However, these agents demonstrate extraordinary competence in coping with and utilizing their environments to perform sophisticated tasks.

In this paper we present three examples of this expert level problem solving performance. Our tasks, however, will be performed by computer-based problem solvers: a robot exploring and mapping its world, a computer-based diagnostic reasoning system for finding failures for discrete component semiconductors, and a control system for a particle beam accelerator. We will describe each of these computational problem solvers in sufficient detail to give an appreciation for how the tasks are addressed, and the sense in which they can be considered solved.

The approach we have taken in these tasks is based on what we call exploration and model building with iterative refinement. The supporting intuition here is simple: the problem solving agent, through exploring its domain, makes a preliminary *model* of its problem solving situation and then refines or clarifies that model as it discovers new relevant information in the process of accomplishing its tasks. A number of questions arise immediately. What is the nature of this so-called model? What are its salient

George F. Luger, PhD
Department of Computer Science
University of New Mexico
Albuquerque, NM 87131 (USA)
E-Mail luger@cs.unm.edu

dimensions? How does a model change or mature in the process of its use? How can a model be seen as good enough to satisfy the constraints of the task at hand? We answer these questions in the contexts of each of our examples.

There is, we feel, an important generalization of our research approach. One may ask the question of how any agent, whether insect, bird, primate, human primate, or computational, interacts with and uses its environment to survive, including an integration into its social structure. We do not claim that our approach is valid across all the multiple forms of intelligence we might envision, but rather that it does offer a possible response to the 'epistemological' problem of how an agent might be able to know, interact with, and survive in its world.

Epistemology, taken from the Greek word επιστημε (episteme), is the study of the nature and use of knowledge or of how an agent can comprehend and is able to survive within its environment. Epistemology includes the very simple issues, including the existence of an extra-subject 'outside' world, the reality of that 'outside' world, including the reality of other agents, and the use of both the world and other agents to support its survival. Epistemological 'access' asks in what sense an agent can comprehend how it knows its world. How or in what manner is the world, in fact, knowable? Is it directly and immediately perceived and just 'out there'? What is the reality of an agent's thoughts about knowing its world? And thus, what is the status in 'what is real' of an agent's thoughts about its thoughts about its world?

These are exciting questions, and most appropriate for our understanding of intelligence whether human, nonhuman, or computational. Before we address these issues further, we would like to present three of our research group's projects in computational intelligence. We feel that the model building and refinement approach we have taken in each of these projects addresses the issue of epistemological access as well as supports the viability of a constructivist epistemology, but we will wait for the final section of this paper to discuss these issues further.

Three Examples of Computational Intelligence

We next present three research projects in computational 'intelligence'. There are several general questions that each of these projects addresses. These include how and what external information does the agent perceive, what aspects of the perceived information are interpreted and encoded for further use by the agent, and how can this agent information be said to be good enough for its further use in problem solving?

The Robot Agent for Exploring and Mapping

The traditional [before 1990; see Brooks, 1989] approach that the artificial intelligence community has taken to robotics was to create for the robot-based computer problem solver a detailed description of its environment. Thus, the robot designed to move around a room might have the length, width, height, color, and location of tables, chairs and other entities in its world. It would have an exact location for walls and doorways, for obstacles to avoid, as well as for the important parameters of the tasks it was to perform. This world was often represented by the predicate calculus (a mathematics-based description language), with the logic of its moves determined by a theorem prover [see the SHAKEY research and STRIPS; Fikes and Nilsson, 1971; Fikes et al., 1972].

The knowledge that drove these early systems was precise, and as a result, often unforgiving. These AI robot designers took a distinctly 'rationalist' approach, with complete detail for the problem domain worked out in an *a priori* and pre-interpreted fashion, based on the often anthropocentric biases of its designer. When it was acknowledged that the very actions of the robot could change – and even, by happenstance, perturb – the environment it which it operated, new frame axioms, non monotonic logics, and truth maintenance systems [McCarthy, 1977; Doyle, 1979] were introduced in an attempt to 'save' this rationalist perspective of the world.

In the early 1990s, Brooks' [1989, 1991] research at MIT offered an alternative computational model for robots, the 'subsumption' architecture. In this approach, Brooks layered together independent systems for interacting with the world. Figure 1 offers an example of such a system, where an *object avoidance* layer supports higher-level competencies for *exploring* an environment and *planning* tasks within it. Brooks claims that the approach of his system is to 'wire finite state machines together into layers of control. Each layer is built on top of existing layers. Lower level layers never rely on the existence of higher level layers'. For Brooks, this offers the possibility of a control system without representation. In fact Brooks claims, 'When we examine very simple levels of intelligence we find that explicit representations and models of the world simply get in the way. It turns out to be better to use the world as its own model' [Brooks, 1991].

Brooks' approach of 'intelligence without representation' is certainly at the opposite extreme of the epistemological spectrum from its logic-based predecessors pre-

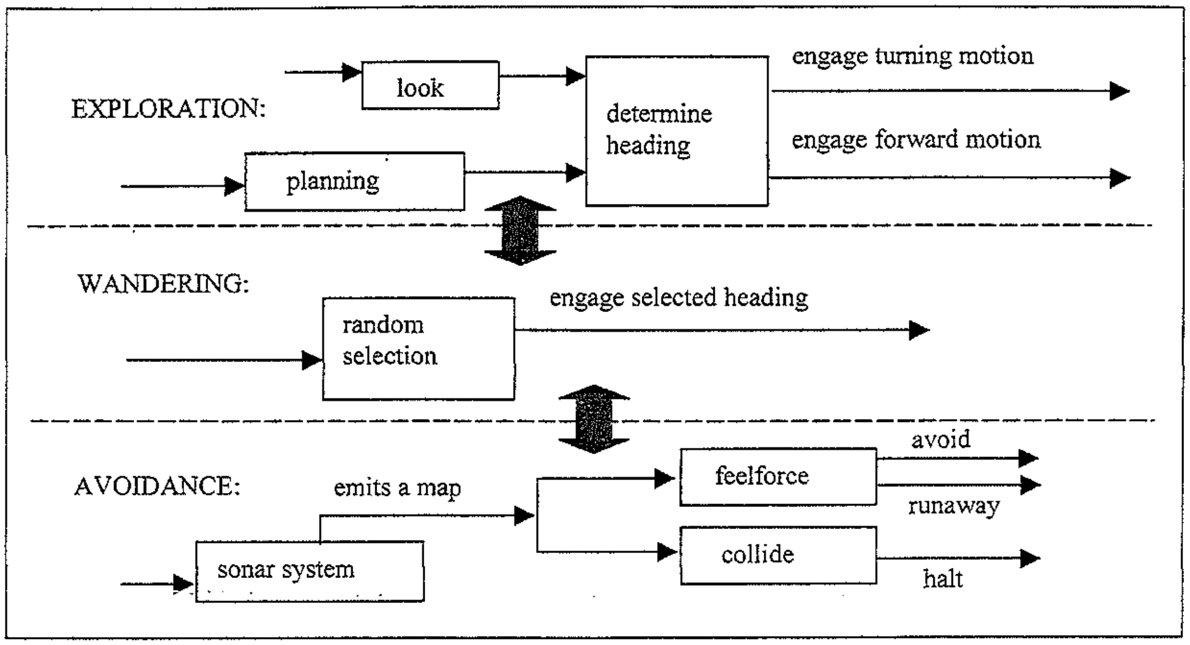


Fig. 1. The interactions within and between layers of the subsumption architecture generate an overall reaction to sensory input from the environment [from Lewis, 2001].

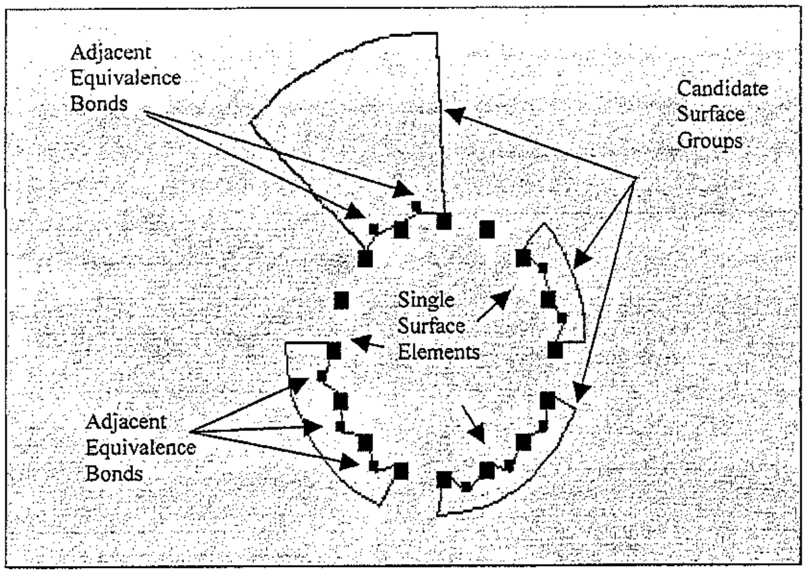
sented earlier. Unfortunately, Brooks' and others' more recent work has not born out the earlier promise of this approach [Brooks and Stein, 1994; McGonigle, 1998; Lewis and Luger, 2000]; and yet it is important to appreciate what contributions were made. Certainly we can see that intelligence is dynamic and operating in a fluid domain. We also know that the world offers *scaffolding* [Clark, 1997] for operations where tasks are indexed and partial solutions can be recorded and saved for later use. But even these aspects of problem solving would seem to require their correlated representational components for the agent.

Several research groups [Moravec, 1988; Thrun, 1998, 2000; Lewis and Luger, 2000; Lewis, 2001] are now offering hybrid representational schemes that can, through interaction with a domain, capture and represent invariances within that context. We call our project 'Madcat' [*mad* from the trade name of the No*mad* robot, and *cat* because we have extended the earlier Copy*cat* work of Mitchell, 1993]. Madcat is a hybrid representational scheme for controlling a robot; the Madcat architecture produces an evolving representation that maps out its world as it explores that world for later use in problem solving.

Madcat is an example of a *cognitive architecture* which means that it is an attempt to capture the full cognitive functioning of an agent [Luger, 1994]. This functioning includes the input of its perceptions, through their interpretation and integration into the agent's memory structures, as well as the further use of this acquired information/knowledge to achieve the agent's goals. Thus, the task in building a cognitive architecture is to create, in software, a fully functioning cognitive agent, in our case, a robot. A number of other cognitive architectures, including those of SOAR [Newell, 1990], ACT* [Anderson, 1983], and Copycat [Mitchell, 1993; Hofstadter, 1995] have preceded the Madcat work [Lewis, 2001].

The Madcat perceptual system includes a set of sixteen sonar sensors fixed at uniform distances around its circular base. Figure 2 presents the Madcat robot with its sixteen sensors and an example set of its input snapshots. The processing of these snapshots produces an evolving representation called the *mapnet*. The *mapnet* (figure 3) reflects both the robot's perceptions at a particular time combined with its *a priori* expectations of patterns within and between these sets of perceptions.

The full Madcat architecture is presented in figure 4. Sequences of snapshots like those of figure 2 are integrated over time within the *workspace*. The relationships within elements of each snapshot as well as the correspondences of relationships between consecutive sets of snapshots are identified in this *workspace*.



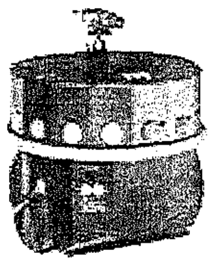


Fig. 2. Representational structures develop around input measurements from a single snapshot with the sonar sensors [from Lewis, 2001].

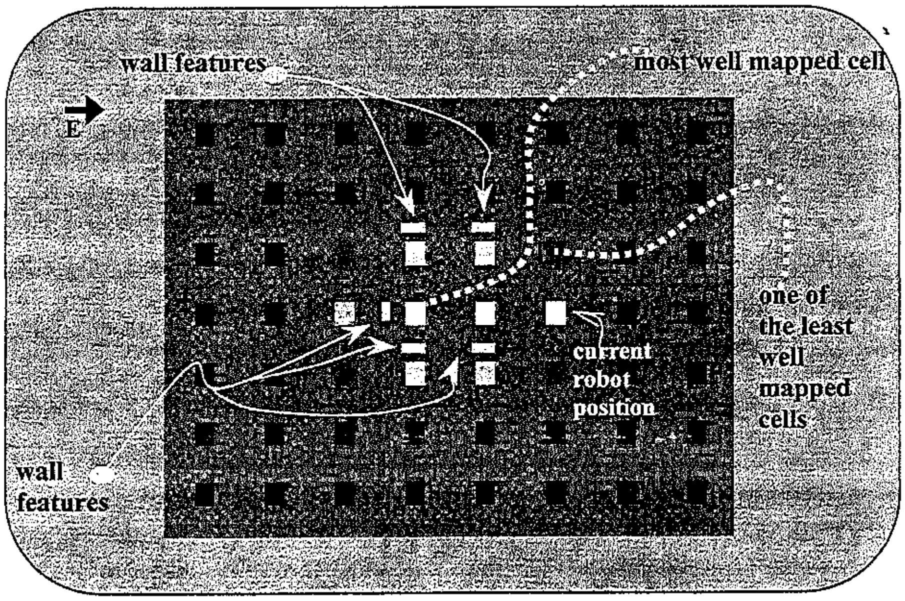


Fig. 3. Mapnet features, corresponding to perceived surfaces in the environment, are constructed based on patterns among structures within and between snapshots [from Lewis, 2001].

The *coderack* contains the pattern matching rules for exploring and directing the overall behavior of the robot system. Currently there are two components of the *coderack*, the first containing rules to build the relational structures within and between snapshots and the second containing rules to organize movement of the robot itself. The feedback mechanisms of the system select which type of *coderack* codelet is to be used next, depending on the robot's context. For example, when a current snapshot indicates that walls and other surfaces are too close to the robot, appropriate survival mechanisms are chosen. At other times the robot has more latitude to search, as prescribed by the *slipnet*, for interesting patterns in its perceptions.

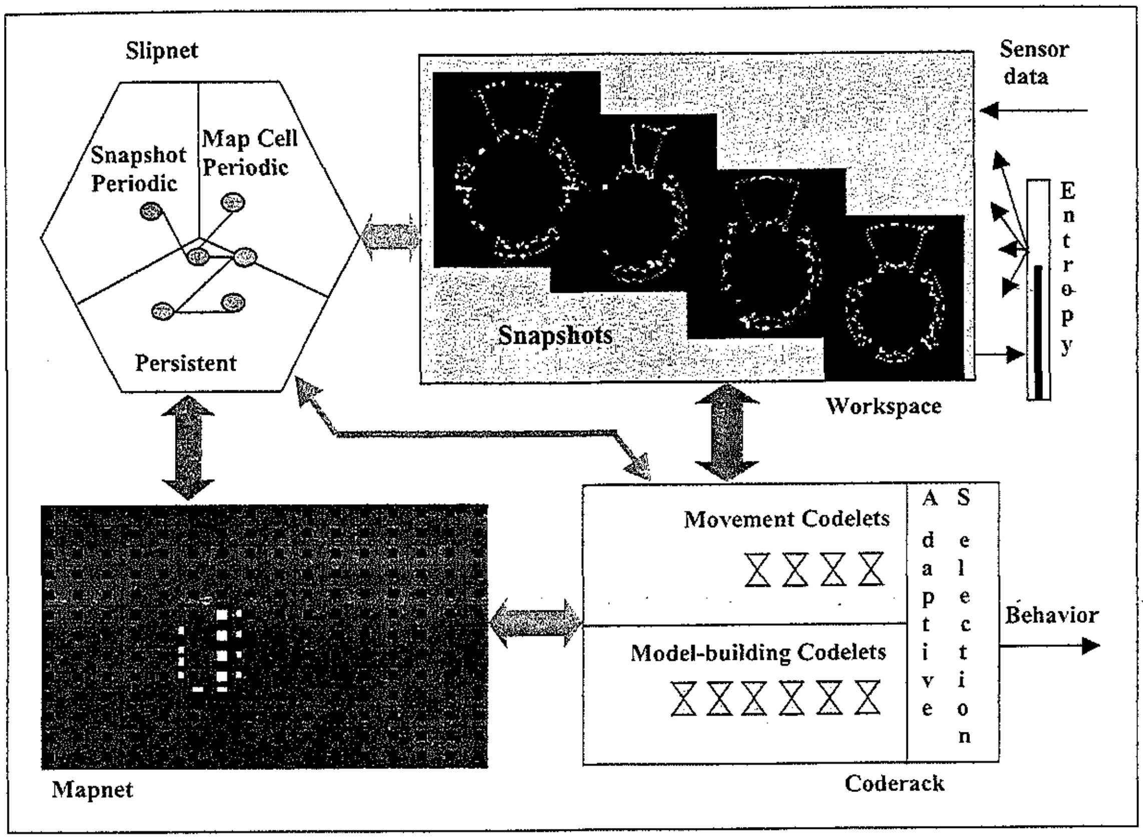


Fig. 4. Organization of the Madcat architecture illustrates the subdivisions of the *coderack* and *slipnet* and how they generate and refine both instantaneous and longer-term models [from Lewis, 2001].

The *slipnet* is divided into three components, each containing rules embodied in an activation-varying network of nodes. These nodes correspond, loosely, to domain-relevant concepts. The first component's nodes, reset with every snapshot, relate the components of the spatial relationships within each individual snapshot. Rules for finding patterns of relationships between consecutive snapshots are embodied in the second component's nodes, which are reset periodically, as the robot moves through the cells of its mapped space. These rules give the system an understanding of possible temporal relationships between snapshots and support the recognition of persistent patterns corresponding to features of the environment. Finally, there are nodes that are persistent and that embody rules for adding features to the developing *mapnet*. These features, in turn, assist in guiding the higher level behavior of the robot (seen in the *coderack* described above).

The *coherence measure* of Madcat reflects how 'comfortably' the current structures in the *workspace* fit together. In an intuitive sense, the *coherence measure* indicates the confidence of the system with its current interpretation of the world, reflected of course, through its attempts to interpret its perceptions through feedback from generated behavior. A close analogue is the notion of entropy in physics, the measure of the disorder of a system. The *coherence measure* can be used to mediate the delivery of goal-driven behavior, reflecting the notion of 'satisficing' as originally proposed by Simon [1969]. That is, it offers a measure according to which a system can decide, in the context of its full set of needs and processing capacities, when a possible action is good enough for accomplishing a task.

Once Madcat explores its world situation, organized and enabled by the rules of the *coderack* and *slipnet*, the *mapnet* reflects and encodes the results. As an example, the grid patterns in *mapnet*, after the successful navigation of a complex corridor, can be seen in figure 5.

To summarize, the Madcat robot is able to create a tentative map of its world as it explores that world. The snap-

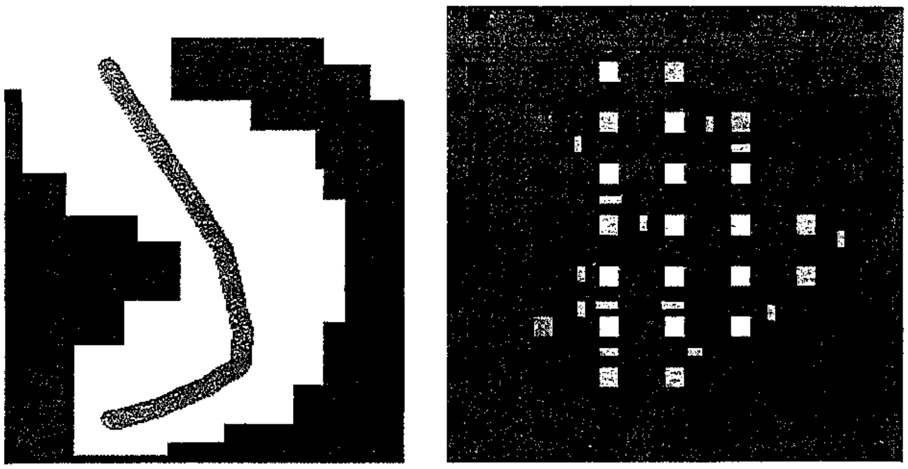


Fig. 5. An environmental configuration provides the goal-directing feedback for the robot; and the generated *mapnet* features demonstrate a sparse representation just sufficient to produce smooth navigation [from Lewis, 2001].

shots of input sonar signals are compared and similar structures linked together by the rule patterns of the *slipnet*. The application of these rules is mediated by the *coderack* as appropriate for each situation. The system's *coherence measure* indicates how satisfied it is with its current interpretation of its domain. A natural extension of this approach would be to use the *coherence measure* to modulate the goal driven activity of the robot agent, i.e., to indicate when a particular interpretation is good enough for its goal driven purposes. These purposes might include determining, for example, whether a structure is suitable for following (a wall) or is sufficient for going through (an entryway), as it moves around its domain. Thus, the good enough measure can organize the system's current goal within the larger context of multiple goals, including object avoidance, exploration, and even recharging its batteries.

A Diagnostic Expert

One view of the exploring robot of the previous section is to see it as an agent trying to determine minimum information/knowledge for mapping out and getting around in its world. This would be much like a blind person exploring unknown territory using very limited sensory input, for example, a cane to 'feel' obstructions. There would be no easy recognition of things in the every day world such as walls, corners of a room, or doorknobs. These fixtures of a normal world for humans must be 'discovered' and 'cataloged' for later use by the robot. Our second example is a further instance of exploratory reasoning, this time in the task of diagnosing and explaining failure patterns of discrete component semiconductors.

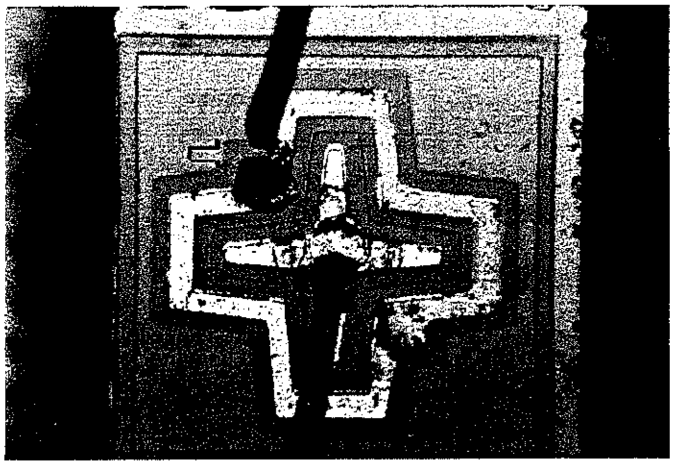


Fig. 6. An example of a semiconductor failure, indicated by the arrow. The failure mode is an electrical 'open', the type of failure is a 'fracture', and the explanation is some type of 'mechanical stress'.

Failure explanation requires an expert not just to note that some aspect of a device is not as it should be, but also to explain how it got that way. Figure 6 presents such a failure. In this situation there is an electrical break – an 'open' – indicated by the arrow. The cause of the break is a broken wire. The type of break is a fracture. The cause of the fracture is most likely mechanical stress. We can now ask how this mechanical stress might have occurred. An answer might be from the improper packing of the device or from the connecting of it into other components, or through some other human mishandling.

Human experts are trained to make these types of diagnoses. The usual training includes several courses in electronics and their associated lab sessions. The expert diagnosticians in the electronics domain also often have years of experience seeing and testing actual failed devices. As with human doctors, the final levels of expertise are achieved only through much practice-based experience.

We have built two computer-based solutions for explaining failures in situations such as that of figure 6. These systems were built after many hours of interviewing human experts in the area of failure analysis for discrete component semiconductors (Discrete component semiconductors are individual devices, such as a transistor or resistor, that are linked together to perform some function, such as to add numbers.) The first system was in the form of a traditional rule-based expert system [Stern, 1996; Stern and Luger, 1997]. This system was intended to assist the human expert in explaining failures and searched through sets of related explanations based on the data it encountered.

The traditional rule-based system is made up of (often hundreds of) related *if – then* rules. For example, *if* you see pattern X *then* look for pattern Y. *If* you find pattern Y *then* check possibility Z. These individual rules, as they are used, will 'chain' together. In the example above, when the expert sees pattern X, and as a result looks for and finds pattern Y, then possibility Z is checked. Thus patterns of data are noted in the failure situation, which direct the organized search through different lines of reasoning that can lead to different possible solutions. This type of reasoning, often called 'abductive', goes from different sets of observable pieces of data to determine the best possible explanation for that data [Peirce, 1958].

Actually, the relationship between data and explanation is very much two-way. The input data shapes the search in terms of possible explanations, while the particular explanation under consideration shapes the search space by suggesting possibilities for acquiring new data. When human experts search for explanations whatever they see shapes what they expect, while at the same time what they expect drives the search for and interpretation of new pieces of data. All this may be captured in a carefully designed rule system.

Our approach to diagnosis also demonstrates how one piece of information can be interpreted differently in the contexts of different explanation scenarios. This situation is captured in figure 7, where individual pieces of data can support different explanations, and at the same time support of the explanation contexts can suggest/require obtaining different new pieces of information. In figure 7 the data of 'traffic slowing down' can be interpreted as either 'road construction' or 'accident' ahead. Further data (seeing orange traffic control barrels or flashing lights) can suggest one of these explanations as 'best' [Luger, 2002].

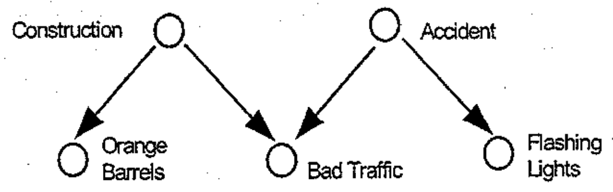


Fig. 7. A representation belief patterns for the 'Bad Traffic' problem.

Our expert diagnostic system was built with many hours of cooperation from human diagnostic experts. The search paths that the program took were intended to be much the same as those the well-trained human expert took in looking at these same problems. In its usual mode of operation it was used alongside the expert, suggesting possible explanations of data, as well as proposing new pieces of information that should be sought to confirm possible explanations. Furthermore, the program could be seen as a methodology for preserving aspects of human skilled problem solving, when the human diagnostic expert might no longer be available.

Our second attempt at capturing the expertise involved in the diagnosis of discrete component semiconductors was with a Bayesian belief network (Bbn). An example Bbn for 'leaking transistors' can be seen in figure 8.

In this belief network we took some of the same information captured in our rule system and gave it a different representation. A Bayesian belief network is an example of a stochastic reasoning system. The nodes in figure 8 represent several aspects of a diagnostic situation. First there are the data nodes, the clear nodes of figure 8, that reflect the likelihood of a piece of data being present or absent in a certain situation. Second, we have nodes for possible explanations, the shaded nodes of figure 8. Finally, in some Bbns there can be 'utility' nodes that reflect the cost of different lines of reasoning. For example, if acquiring one piece of data has a very high cost, the expert system might not use that line of reasoning until other approaches have already been explored.

In a stochastic model, the nodes of the system are linked together by a measure of how often the phenomena they represent occur together in the natural world. In figure 7, for example, the 'traffic slowdown' might happen for an 'accident' 0.75 of the time and for 'road work' 0.25.

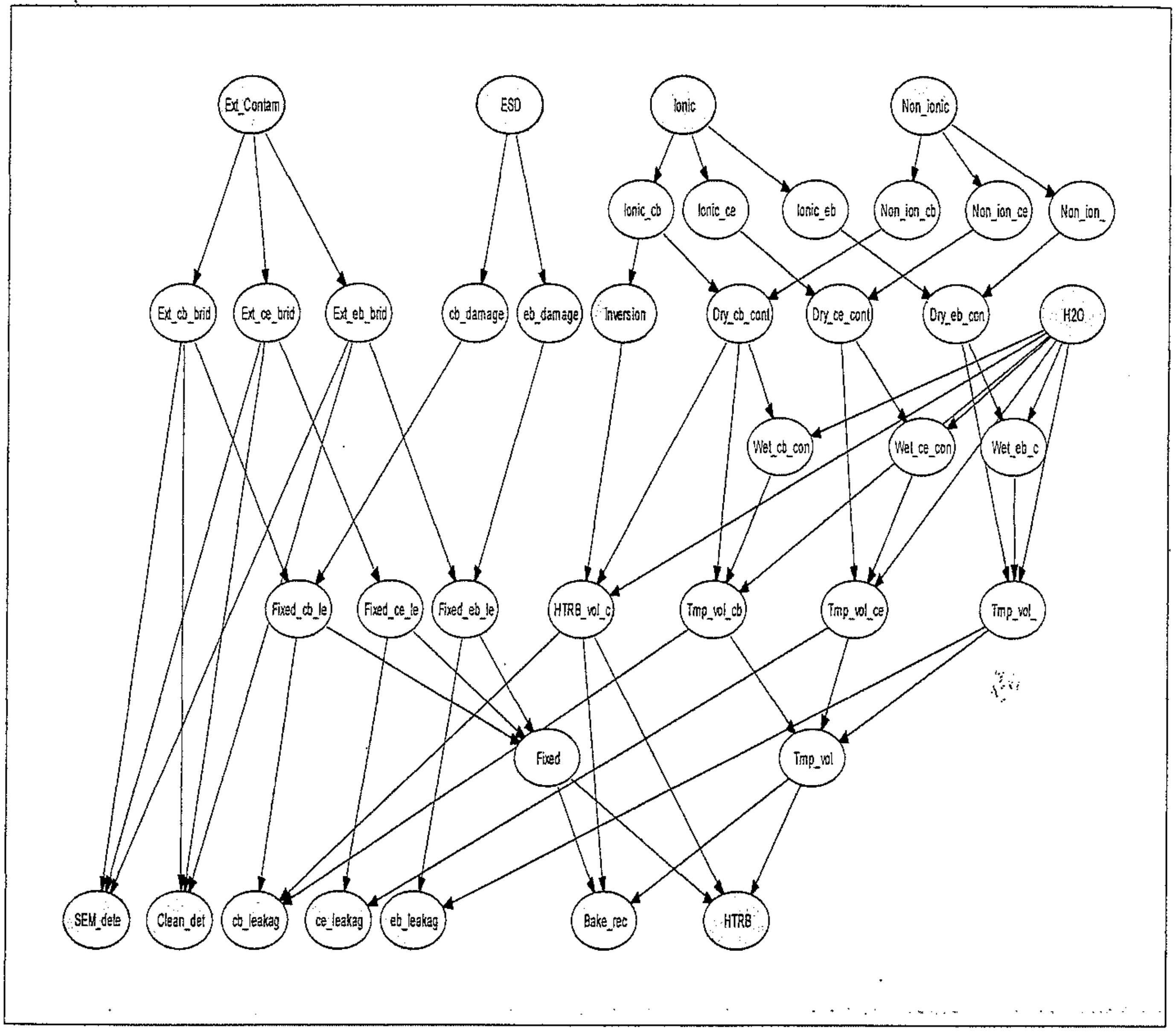


Fig. 8. An example Bayesian belief network for 'leaking transistors'. Clear nodes represent data and shaded nodes possible explanations. The direction of the arrow indicates the relationship between pairs of nodes.

In a similar fashion all the nodes in the Bbn are linked by a measure (the *a priori* probability) of how often the phenomena they represent happen together in the natural world. In this sense the Bbn representation need not be created by the collected knowledge of several human experts, rather it is simply the reflected measures of co-occurring data, regardless of how these relationships are captured. In a natural sense, the Bayesian network reflects what the expert sees and expects when looking at a problem situation. The full sets of collected information and their co-relations offer a snapshot of the world as it is.

As described so far the Bbn offers a static view of the natural world; but its power extends much further. First, when new pieces of information are acquired (*a posteriori* knowledge) and added to the Bbn representation, the co-relations in the network change to reflect the best explanations for this new information. For example, in the situation of figure 7, the presence of flashing lights changes our

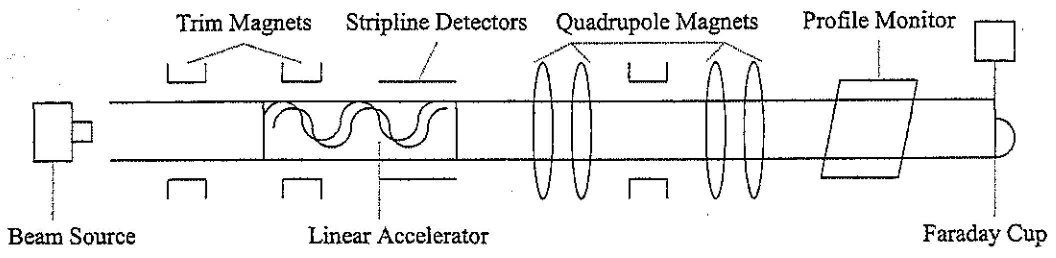


Fig. 9. A graphic representation of an accelerator beamline. The magnets steer and focus the beam by guiding and changing the direction of the particles. Monitors, such as the Faraday cup, measure the beam's strength and profile.

expectation of an accident. Second, when new goals are identified, the system indicates which pieces of new information are the most important to consider for support of these goals. Finally, and our current system does not yet do this, the Bbn representation is capable of learning new co-relations of the data as it acquires further experience within its application domain.

In this section we have presented two different computational models for representing expert human diagnostic problem solving performance. The first, the rule-based expert system, is often used to create an explicit web of reasoning rules that reflect the arguments and decisions taken by the human expert. These rule systems are usually constructed through many hours of interviewing and testing in conjunction with the human expert. Even though these systems can be 'brittle' in that they sometimes misrepresent the precise data observations and calculations of the expert, they can be further calibrated as their failures are observed. This progressive approximation seems to reflect how humans come to understand a problem domain. Thus, rule systems can offer a very powerful approximation of human expert performance [Luger, 1994, 2002].

The Bayesian belief network, either through direct learning of the co-relations in a problem domain or through explicit programming, comes to reflect the *a priori* expectations of the 'causal' relations of the problem situation. As new data are added, or the support for some goal is queried, the dynamic reconfiguration of the network reflects causal contingencies in the natural world, a characterization of how things 'are'. In our continuing research in this area we are creating a general representational scheme for capturing stochastic relationships as well as general purpose inferencing algorithms to capture and calibrate new *a posteriori* information. We call our general purpose representation language a *stochastic lambda calculus* [Pless et al., 2000; Pless and Luger, 2001].

Representing Complex Knowledge and Skill: A Control System for Particle Beam Accelerators

Human experts are remarkable for developing sophisticated control behavior for operating in complex situations, such as handling heavy machinery, driving cars, or flying airplanes. Over the past six years one of our research groups has designed, built, and tested a portable, intelligent software system for accelerator beamline tuning. Our approach to this problem has been general in that our software is re-configurable to a wide variety of particle beam accelerators. Our control architecture has a multiple layer and hierarchical organization in which knowledge-based decision making is used to re-configure lower level optimization and control decisions [Klein et al., 1999, 2000]. In many aspects this control software replicates the skills of highly trained particle beam control physicists.

A particle beam accelerator is a device that is used to transport highly charged particles from a source to a target. The beamline consists of a number of elements designed either to change the beam's characteristics (direction, size, shape, etc.) or in some manner to monitor these characteristics. The purpose of the beamline is to steer, focus, and otherwise modify the beam so that it is transported through the beam pipe to a specified location all the while maintaining its characteristics within an acceptable range. The final beam should reach the target with a specific set of characteristics, as determined by the work being done. Figure 9 shows a simple accelerator beamline which includes trim magnets for steering, quad-

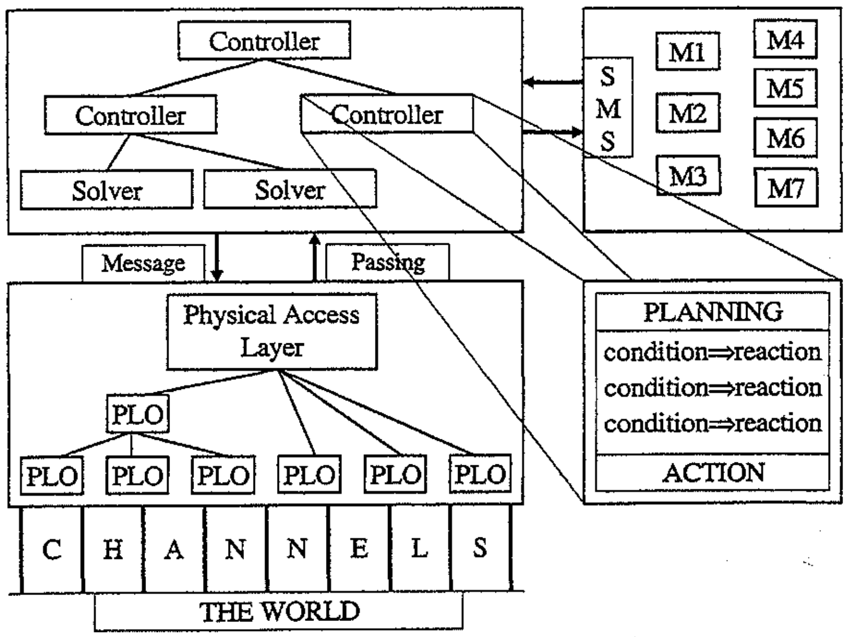


Fig. 10. An object-oriented view of the accelerator control problem. The lower left component of the figure represents the 'objects' that capture the many pieces of information on the channels of the high-speed data bus that monitors the actual components of the accelerator hardware. The upper and rightmost components of the figure represent the model building and planning modules of the software.

rupole magnets for focusing, Faraday cups and stripline detectors for measuring current, and profile and popup monitors for measuring the size and position of the beam [Klein, 1997; Klein et al., 1997a].

Accelerator beamlines are designed by placing various components along the beam pipe to produce specific effects. A good design will minimize the number of components necessary to maintain acceptable beam conditions while still allowing enough freedom of control to achieve a range of target conditions. Unfortunately, actual systems rarely work exactly as designed. Problems arise from imperfect beam production, remnant magnetic fields, poorly modeled beam behavior, misplaced or flawed control elements, and changes to the design and use of the facility after it has been built. Even with many built-in diagnostic tools such as beam profile monitors and current detectors, the uncertainty of each situation can make beamline control difficult.

Although most modern accelerator facilities have some automated beamline tuning, normal operation still requires manual intervention. Human control experts need to generate the initial start up tune of the beam. Sometimes, the non-expert controller will start the system and then hand the tuning task over to the expert physicist for fine calibration. The manual task of generating a proper tune can be lengthy, taking many hours, if not days.

We began our project in particle beam control by watching the human experts perform this task. We were able to visualize with them the various models from physics they used to understand the data from the beam system. We also observed them revising these models as the system's output changed over time. Our approach to building software for the accelerator control problem was to employ the traditional object-oriented technology to represent the objects in the physical world of accelerators as software objects in our program. This allowed us, for example, to create classes such as 'magnets', with subclasses of 'trim magnets'. Particular instances of trim magnets would make up the entities that constituted the full accelerator system of figure 9. In object-oriented technology, the general properties that make up a type of entity, magnets say, are brought together in a magnet class, while the specifics of these classes are brought together into one instance component. These specifics would include the parameters of a particular entity, including its location, power, etc. as well as its specific function in the overall beam focusing scenario. An object-oriented view of the accelerator tuning system can be seen in figure 10 [Klein, 1997; Klein et al., 1999].

At this level of abstraction, our control system allowed us to reason out the beamline tuning problem by thinking about how our 'software objects' fit together and interacted. We were no longer faced with thousands of individual pieces of data that had to somehow be fit together within a large program: rather we had software objects, similar to the physical objects of figure 9, that fit together in specific ways to make a coherent picture. Our model allowed our software to act as expert physicists might in solving the tuning problem.

An important result of thinking about the tuning problem from this object-based viewpoint was the ease of software component reuse. We were able to take a software product that required several person months to build for use at one accelerator facility (the Brookhaven AFT) and to rebuild it in less than a person-week for use on another beamline (the Argonne ATLAS facility). All that was required was to reconfigure the existing objects to the particular situation at Argonne, and to add several new objects that were particular to that new facility [Klein et al., 1997a, b, 2000].

Our object system was also appropriate for creating and testing various software models of a physical system. The top-down hierarchical organization allowed us to make hypothetical conjectures about a physical system and then to test whether or not this hypothesized system was actually reflected in the physical system we were encountering at the time. This was a great assist for physicists trying both to bring a beam into focus and/or attempting to fine tune the resulting system. A *planning* component of the system also supported the dynamic configuring and testing of an accelerator system. A representation of the hierarchically organized planning and model-based views of our software can be seen in the top and rightmost components of figure 10.

The *planning* component of the software allows the human expert to generate planning sequences, in which the computer organizes a sequence of actions that can accomplish a specific task. For instance, if the focus of the beam is to have a certain intensity at a specific location, a particular plan can attempt to accomplish that task. The *model* component of the software allows the human user to understand the parameters of the system in the context of a particular interpretation or model. For example, allowing the expert to ask what a particular arrangement of parameters might mean for the system. This supports the continued adjustment of the system to accomplish specific tasks.

An exciting result of our model theoretic approach was the task of trim magnet location at the Argonne ATLAS facility [Klein et al., 1997b]. Because of time, use, and the changing parameters of the Argonne facility itself, the exact location of one of their tuning magnets was unknown. The fact that the precise location of a magnet weighing several hundred pound could not be determined is not as impossible as one might at first think. In fact, many of these magnets are not accessible, buried beneath the facility, and many of them actually change their location, power, and field strength over time.

With our model refinement algorithm we were able to re-establish the location and power parameters of this magnet. Our model-based approach simply asked, over repeated trials on the beamline, what model (or organization of components) could account for the observed behaviors; and in a relatively short time we were able to determine the magnet's parameters. What is really interesting to consider in this situation is that we might not have actually found the exact location of the magnet at all; but for all the practical purposes represented by our set of experiments and models, this location was a good enough fit. This is an important issue in the context of any intelligent agent attempting to understand and use its world: what is 'really out there' and in what sense can/do we know and use 'it'.

Finally, Stern and Lee [1999] have extended this model refinement approach with their work at the Stanford Linear Accelerator Center. In what they describe as *model calibration* they are able to refine through accelerator use current supposed models to get a more precise fit to the accelerator system; and calibrating the accelerator itself makes it fit more closely to the physicists needs and expectations for that system.

In this section we have seen several examples where dynamic software models are used in the context of problem solving. With accelerator beamline tuning we considered the situation of a complex domain, where many of the parameters are not directly observable and many components interact in a highly non-linear fashion. A possible model for a situation allows the expert to 'test' the real: 'Is this the way things are, and if not, how can I adjust my model for a better fit?' In the final section of this paper we propose model building and refinement as a critical epistemological construct.

Towards a Constructivist Epistemology

We have presented three examples of computational intelligence. The goal in all three systems was to design support architectures for agents' better understanding and use of their environments. In each case this required a representational scheme, the *mapnet* for the robot, the stochastic reflection of semiconductor failures for diagnosis (the Bbn of figure 8), and a set of models of the physics of the accelerator for tuning (figure 10). In each case these representational schemes were sufficient to capture the invariances of the problem domain for use in an agent's search for solutions.

These representations are not at all similar to the more traditional approaches of the AI community that used logic, frames, or other representational schemes. These traditional approaches usually reflected the pre-interpreted and *a priori* worldviews of their system designers. In our models, the commitment to an interpretation of 'external' phenomena, with its associated 'confidence measure' is made by the agent as it discovers and explores its world. Our evolving representations are a reflection of the embodied agent exploring and using its world for its own goals and purposes.

In support of this notion of the task-based or goal-driven agent, each problem representation also possesses a comfort or confidence measure. This measure is represented by the *temperature* of the Madcat system, the correlation measures and solution confidences in stochastic diagnosis, and the model refinement procedure of the accelerator beam tuning algorithms. These measures reflect the confidence or 'belief' that a system has with its current view of the world, and is a critical support for good enough problem solving. In fact, an agent rarely – if ever – has perfect confidence in its perceptions and their interpretation. For all agents in domains of mixed and multipurpose goals, it is critical to have a prioritization scheme that supports both the continued acquisition as well as the eventual use of information.

But there are further issues important here, namely the design philosophy that supports our approach to an agent exploring, learning about, and using its world. This topic comes under the general assumption of an epistemology that enables and supports an agent's interaction with its world. Our epistemology is *constructivist*.

Long ago, Plato, in the words of the slave Meno, posed the problem of how an agent is able to explore and learn new knowledge about its environment:

And how can you enquire, Socrates, into that which you do not already know? What will you put forth as the subject of the enquiry? And if you find out what you want, how will you ever know that this is what you did not know [Plato, 1961]?

Meno's point is important, and not easily dismissed. What IS something that you don't know? And if you don't know a thing now, how can you ever recognize it later? Plato answered this question with his ideas on reincarnation and remembering. A constructivist addresses these issues by bringing to them modern concepts from psychology, philosophy, and computation.

A constructivist epistemology hypothesizes that all understanding is the result of an interaction between energy patterns in the world and mental categories imposed on the world by an intelligent agent [Piaget, 1954, 1970; von Glaserfeld, 1978]. Using Piaget's descriptions, we *assimilate* external phenomena according to our current understanding and *accommodate* our understanding to the 'demands' of the phenomena.

Constructivists often use the term *schemata* to describe the *a priori* structures used to organize experience of the currently present external world. This term is taken from the British psychologist Bartlett [1932] and its philosophical roots go back to Kant [1781/1964]. In this viewpoint, observation is not passive and neutral, but active and interpretative. In our examples of computational problem solvers the role of 'schema' is taken by the *mapnet*, the Bayesian belief network, and the models supporting beamline tuning. These schemas are reactive and evolve because of their links and interactions with other system components as well as the 'outside' world.

Perceived information, Kant's *a posteriori* knowledge, seldom fits precisely into our preconceived and *a priori* schemata. From this tension, the schema-based biases the subject uses to organize experience are either modified or replaced. The need for accommodation in the face of unsuccessful interactions with an environment drives a process of cognitive equilibration. Thus a constructivist epistemology is fundamentally one of cognitive evolution and refinement. An important consequence of constructivism is that any interpretation involves the imposition of the observers' concepts and categories on reality.

When Piaget [1954, 1970] first proposed a constructivist approach to understanding, he called it *genetic epistemology*. The lack of a comfortable fit of the present schemata to the world 'as it is' creates a cognitive tension. This tension drives a process of schema revision. Schema revision, Piaget's *accommodation*, is the continued evolution of an agent's understanding towards *equilibration*.

Schema revision and continued movement towards equilibration is a genetic predisposition and accommodation of a system to the structures of society in an 'external' world. It combines both of these forces and reflects an embodied predisposition for survival. For humans, schema modification is both an *a priori* expression of our genetics, as well as an *a posteriori* function of society and the world. It reflects the embodiment of a goal-driven agent, of a being in space and time.

There is a blending here of the empiricist and rationalist traditions, mediated by the goals of agent and societal survival. As embodied, agents can comprehend nothing except that which first passes through their senses. As accommodating, agents survive through learning the general patterns of an external world. What is perceived is mediated by what is expected, what is expected is influenced by what is perceived. These two functions can only be understood in terms of each other, and both of them are modulated by the exigencies of survival, of a 'good enough' response.

Furthermore, this accommodation is seldom conscious or rational, nor are agents often aware of the schemata that support interaction with the environment: rather it is the spontaneous response of an embodied and goal-driven system. Accommodation is constitutive of equilibration with the world, it supports learning and all adjustments to an environment, but it is rarely a perceptible element of conscious mental life. In fact, as sources of bias and prejudice both in science and society, we are more often than not unaware of the content of our *a priori* schemata.

Finally, why is a constructivist epistemology particularly useful for addressing the problems of understanding intelligence itself? How can an agent in an environment understand its own understanding of that environment? We believe that constructivism addresses the epistemological access problem in philosophy and psychology. For well over a century, there has been a struggle in both these disciplines between two factions, the positivist, which proposes to infer mental phenomena from observable physical behavior, and a more phenomenological approach which allows the use of first-person reporting for access to cognitive phenomena. This factionalism exists because both modes of access to psychological phenomena require some form of inference or construction.

In comparison to physical objects, which are often naively thought to be 'directly accessible', the mental states and dispositions of an agent seem particularly difficult to characterize. In fact, we contend that this dichotomy between direct access to physical phenomena and indirect access to the mental is illusory. The constructivist analysis shows that no experience of things is possible without the use of some model or schema for organizing that experience. In scientific inquiry, as well as in our normal human experiences, this implies that all access to phenomena is through exploration, approximation, and model refinement.

Theories are like nets; he who casts, captures.
L. Wittgenstein [1953]

Acknowledgements

I wish to thank the Krost Symposium for inviting this presentation. I also thank a generation of PhD students, especially Bill Klein, Joseph Lewis, Dan Pless, Carl Stern, and Bill Stubblefield whose dissertations embodied aspects of this constructivist methodology. I am indebted to the National Science Foundation (NSF 9800929, NSF 9900485) and NASA (NASW-00043) for support in this research. I thank the Computer Science Department of the University of New Mexico and the Laboratory for Cognitive Neuroscience at the University of Edinburgh for the intellectual environments that both enabled and supported a constructivist epistemology (G.L.).

References

Anderson, J.R. (1983) The Architecture of Cognition. Harvard University Press, Cambridge, MA.

Bartlett, F. (1932) Remembering. Cambridge University Press, London.

Brooks, R. (1989) A robot that walks: Emergent behaviors from a carefully evolved network. Neural Comp., *1:* 253–262.

Brooks, R. (1991) Intelligence without Representation. Proceedings of the IJCAI-91. Morgan Kaufmann, San Mateo, CA.

Brooks, R., and L. Stein (1994) Building brains for bodies. Auto. Robots, *1:* 7–25.

Clark, A. (1997) Being There. MIT Press, Cambridge, MA.

Doyle, J. (1979) A truth maintenance system. Artificial Intel., *12:* 231–272.

Fikes R.E., and N.J. Nilsson (1971) STRIPS: A new approach to the application of theorem proving to artificial intelligence. Artificial Intel., *1:* 189–208.

Fikes, R.E., P.E. Hart, and N.J. Nilsson (1972) Learning and executing generalized robot plans. Artificial Intel., *3:* 251–288.

Hofstadter, D. (1995) Fluid Concepts and Creative Analogies. Basic Books, New York.

Kant, I. (1781/1964) Immanuel Kant's Critique of Pure Reason (Trans. by N.K. Smith). St. Martin's Press, New York.

Klein, W. (1997) A Software Architecture for Intelligent Control. Ph.D. Dissertation, Department of Computer Science, University of New Mexico, Albuquerque, NM.

Klein, W., C. Stern, G. Luger, and E. Olsson (1997a) Designing a portable architecture for intelligent particle accelerator control. Proceedings of the Particle Accelerator Conference, American Physical Society.

Klein, W., C. Stern, G. Luger, and E. Olsson (1997b) An intelligent control architecture for accelerator beamline tuning. Proceedings of the Innovative Applications of Artificial Intelligence Conference. MIT Press, Cambridge, MA.

Klein W.B., R.T. Westerveldt, and G.F. Luger (1999) A general purpose intelligent control system for particle accelerators. J. Intel. Fuzzy Systems, *7:* 1–12.

Klein, W., C. Stern, G. Luger, and D. Pless (2000) Teleo-reactive control for accelerator beamline tuning. Artificial Intelligence and Soft Computing: Proceedings of the IASTED International Conference. IASTED/ACTA Press, Anaheim, CA.

Lewis, J.A. (2001) Adaptive Representation in a Behavior-Based Robot: An Extension of the Copycat Architecture. Ph.D. Dissertation, Department of Computer Science, University of New Mexico, Albuquerque, NM.

Lewis, J.A., and G.F. Luger (2000) A constructivist model of robot perception and performance. Proceedings of the 22nd Annual Conference of the Cognitive Science Society. Erlbaum, Hillsdale, NJ.

Luger, G.F. (1994) Cognitive Science: The Science of Intelligent Systems. Academic Press, New York.

Luger, G.F. (2002) Artificial Intelligence: Structures and Strategies for Complex Problem Solving. Addison Wesley, London.

McCarthy, J. (1977) Epistemological problems in artificial intelligence, Proceedings IJCAI-77, 1038–1044, Morgan Kaufmann, San Francisco, CA.

McGonigle, B.O. (1998) Autonomy in the Making: Getting Robots to Control Themselves. International Symposium on Autonomous Agents. Oxford University Press, Oxford, UK.

Mitchell, M. (1993) Analogy Making as Perception. MIT Press, Cambridge, MA.

Moravec, H. (1988) Sensor Fusion in certainty grids for mobile robots. AI Magazine, *9:* 61–74.

Newell, A. (1990) Unified Theories of Cognition. Harvard University Press, Cambridge, MA.

Peirce, C.S. (1958) Collected Papers 1931–1958. Harvard University Press, Cambridge, MA.

Piaget, J. (1954) The Construction of Reality in the Child. Basic Books, New York.

Piaget, J. (1970) Structuralism. Basic Books, New York.

Plato (1961) The Collected Dialogues of Plato (ed. by E. Hamilton and H. Cairns). Princeton University Press, Princeton, NJ.

Pless, D., and G. Luger (2001) Toward general analysis of recursive probability models. Proceedings of UAI-99. Morgan Kaufmann, San Francisco, CA.

Pless, D., G. Luger, and C. Stern (2000) A new object-oriented stochastic modeling language. Proceedings of IASTED International Conference. IASTED/ACTA Press, Anaheim, CA.

Simon, H.A. (1969) The Sciences of the Artificial. MIT Press, Cambridge, MA.

Stern, C. (1996) An Architecture for Diagnosis Employing Schema-Based Abduction. Ph.D. Dissertation, Department of Computer Science, University of New Mexico, Albuquerque, NM.

Stern, C., and M. Lee (2001) Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems '99, Trieste, Italy.

Stern C., and G. Luger (1997) Abduction and abstraction in diagnosis: A schema based account. *In* Situated Cognition: Expertise in Context (ed. by P.J. Feltovich, K. Ford and R.R. Hoffman), MIT Press, Cambridge, MA, pp. 363–381.

Thrun, S. (1998) Bayesian landmark learning for mobile robot localization. Machine Learning, *33:* 41–76.

Thrun, S. (2000) Probabalistic algorithms in robotics. AI Magazine, *21:* 93–109.

von Glaserfeld, E. (1978) An introduction to radical constructivism. *In* The Invented Reality (ed. by Watzlawick), Norton, New York, pp. 17–40.

Wittgenstein, L. (1953) Philosophical Investigations. MacMillan, New York.

59 | 1–2 | 02

Brain, Behavior and Evolution

The Evolution of Intelligence: Brain, Behavioral and Computational Approaches

21st Annual Krost Symposium,
Seguin, Tex., March 22–23, 2001

Guest Editor
Scott Bailey, Seguin, Tex.

S. Karger
Medical and Scientific
Publishers
Basel • Freiburg
Paris • London
New York • New Delhi
Bangkok • Singapore
Tokyo • Sydney

KARGER