Approaches to Artificial Intelligence

Nils Nilsson
David Rumelhart

SFI WORKING PAPER: 1993-08-052

SANTA FE INSTITUTE

Report of a Workshop on

APPROACHES TO ARTIFICIAL INTELLIGENCE

Sponsored by
The National Science Foundation

Conveners: Nils Nilsson and David Rumelhart
Stanford University

Held at the Santa Fe Institute
Santa Fe, New Mexico
November 6-9, 1992

August 20, 1993

1 Introduction

The field of artificial intelligence (AI) has as its goal the development of machines that can perceive, reason, communicate, and act in complex environments much like humans can, or possibly even better than humans can. Even though the field has produced some practically useful machines with rudiments of these abilities, it is generally conceded that the ultimate goal is still distant. That being so, there is much discussion and argument about what are the best approaches for AI—best in the sense of laying the core foundations for achieving ultimate goals as well as best in the sense of producing practically useful shorter-term results. Thus, a number of different paradigms have emerged over the past thirty-five years or so. Each has its ardent advocates, and some have produced sufficiently many interesting results so as not to be dismissable out of hand. Perhaps combinations of these approaches will be required. In any case, the advocates of these approaches often feel that theirs is the "breakthrough" methodology that deserves special support.

In order to acquaint researchers and others with these paradigms and their principal results, the Santa Fe Institute hosted an NSF-sponsored workshop to which advocates actively working in the different approaches were invited. Rather than organize the workshop around the main sub-disciplines of AI, namely machine vision, natural language processing, knowledge representation, planning, expert systems, and so on, we sought to elucidate the different views on how the whole, or major parts, of AI ought to be attacked.

The attendees all felt that the workshop achieved its goal of acquainting leading workers with each others' points of view and the relationships among them. In this report, we will present abstracts of all of the presentations (written by their authors after the workshop) and summaries of the discussions that took place at the workshop (written from rough notes).

It seems to us that the several paradigms and some of their chief spokespeople can be clustered into three or four major, not-necessarily-mutually-exclusive, groups. We devoted an all-day session to each of these four groups of approaches. Each session began with an historical overview followed by other presentations representative of that family of approaches. The purpose of the overview was to put the other presentations in that session into historical perspective. Each of the speakers was asked to highlight both the strong and the weak points of his or her approach and to describe the class of problems for which it is best suited. After each presentation, a discussant offered some initial comments; these were followed by general discussion.

The four major categories of approaches that we considered were:

- Symbol-Processing Approaches

  Newell and Simon's "physical symbol-system hypothesis" [Newell 76] claims that all aspects of intelligent behavior can be exhibited by physical symbol systems, that is by formal manipulation of symbols and symbol structures in machines such as digital computers. This hypothesis is not yet universally accepted, but much of what might be called "classical" AI is guided by this hypothesis. Under this category we included presentations on:

  - State-Space Searching, Richard Korf
  - Declarative Knowledge Bases and Logical Reasoning, Hector Levesque
  - The Blackboard Architecture as a Foundation for Adaptive Intelligent Systems, Barbara Hayes-Roth
  - Soar, Paul Rosenbloom

- Biocomputational Approaches

  There are a number of approaches concerned with machines that do not process symbols as such but, instead, react directly to input signals of some sort. Under this category we included presentations on:

  - Second-Wave Connectionism, Paul Smolensky
  - Genetic Programming, John Koza
  - Behavior-based Artificial Intelligence, Pattie Maes
  - Real-time Learning and Control, Andrew Barto

- Heterogeneous Approaches

  These approaches are usually based on symbol processing but with the added feature of having several symbolic processes working together in some manner.

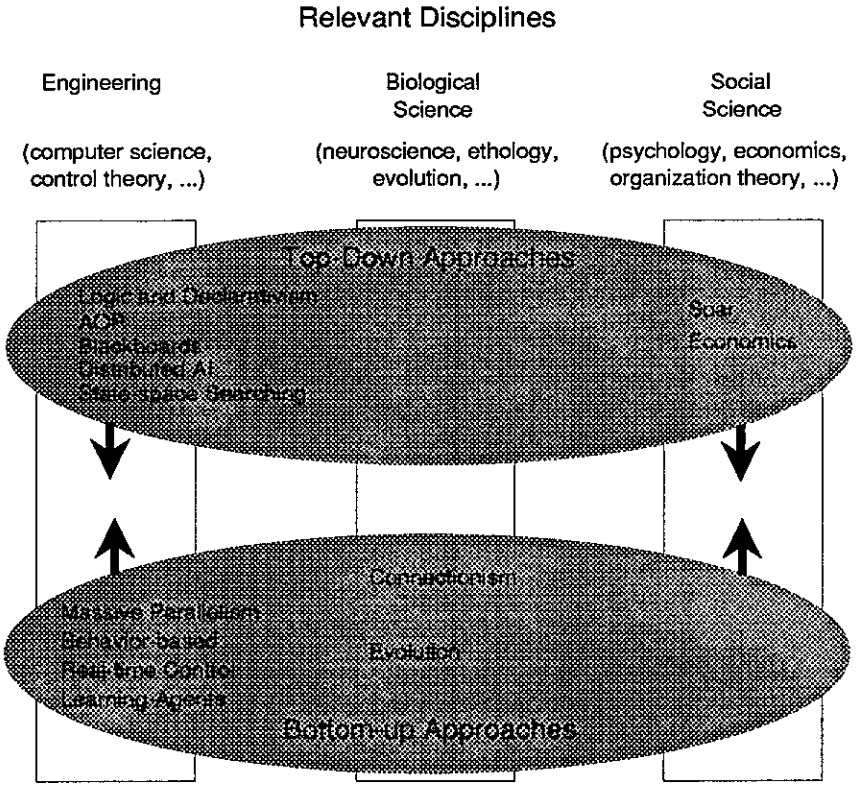
  - Distributed Artificial Intelligence, Victor Lesser
  - Economic Approaches to Artificial Intelligence, Michael Wellman
  - Massively Parallel AI, Dave Waltz
  - Agent-Oriented Programming, Yoav Shoham

- Integrative Approaches

  There are some examples of systems that borrow ideas from several of the approaches discussed in this workshop. We explored two such systems and then discussed the need for and the possibility of a synthesis. Under this category we included presentations on:

  - Learning Agents, Tom Mitchell
  - Integrated Agents, John Laird

During an introductory session at the workshop, Nilsson proposed a related way to organize the different approaches along two dimensions, namely 1) the scientific discipline underlying the approach and 2) whether the approach attempts to arrive at intelligent behavior in a bottom-up manner or in a top-down manner. After reflecting on this scheme while assembling this report, he would place the approaches as shown in the accompanying diagram. At the end of the workshop, Rumelhart proposed an alternative organizational scheme; it is discussed at the end of this report.



2 Abstracts of Presentations and Summary of Discussions

2.1 Session 1: Symbol-Processing Approaches

2.1.1 State-Space Searching

Presentation by Richard E. Korf

Abstract

The first artificial intelligence programs were heuristic search programs, such as the chess programs first suggested by Shannon in 1950 [Shannon 50]. Other early programs were the Logic Theorist of Newell Shaw and Simon in 1956, Samuel's checker player in 1959, Gelernter's geometry theorem prover in 1959, Tonge's assembly-line balancer in 1960, and Slagle's symbolic integrator in 1961 [Feigenbaum 63]. The reason for this is that search is well-suited to these types of high-level reasoning tasks, which typify the classical notion of intelligence. Research in heuristic search has continued for two primary reasons: it is still the best approach for many problem-solving tasks, and it is a vital enabling technology for other approaches to AI, such as theorem proving in symbolic logic.

Work in heuristic search can be classified into three different types of problem domains: 1) single-agent tasks such as theorem proving, robot navigation, and planning, 2) two-player games such as chess and checkers, and 3) constraint-satisfaction problems such as boolean satisfiability

and scheduling. Most of these problems are NP-hard or worse. As a result, while polynomial-time knowledge-based approaches can yield good solutions, better solutions require massive amounts of computation, most often in the form of a search through a problem space. For example, chess machines are very high performance AI systems, relative to humans, and they achieve this primarily through a great deal of search. As another example, while there are many polynomial-time approximation algorithms for the travelling salesperson problem, optimal or near-optimal solutions require a great deal of search.

Heuristic search algorithms can be decomposed into at least five different component parts, each presenting a number of different alternatives to choose from. The first is the problem state representation, which could be partial solutions or complete solutions. Closely related is the choice of operators, witch may extend a partial solution, or incrementally modify a complete solution. Next is the cost function, which could be the actual cost of a complete solution, or a heuristic estimate of the eventual cost of a partial solution. A major component is the control structure of the algorithm, such as breadth-first, depth-first, best-first, simulated annealing, etc. Finally there is the termination condition, such as waiting for an optimal solution, or terminating with a satisficing solution according to some criteria.

Some examples of current research problems in heuristic search include improving the performance of search algorithms in terms of time, space, and solution quality, adapting search algorithms to run on parallel hardware, developing new selective search algorithms that focus their attention on the most important parts of the search space, and search algorithms for on-line problems where decisions are required in real time. The research methodology is typically to develop new algorithms, test them on common domains so they can be compared to existing algorithms, and measure their performance, both analytically and experimentally.

The main strengths of the search approach to AI are its generality, in the sense that most algorithms require little domain-specific knowledge, and the fact that it achieves arbitrarily high performance with increasing computation. The main weaknesses are its computational complexity, particularly for problems with very large branching factors, its psychological implausibility, since people clearly cannot perform large-scale searches, and the fact that most search algorithms do not directly incorporate learning.

For survey articles about search in artificial intelligence see [Korf 88, Pearl 88, Korf 92].

Discussion led by Paul Smolensky and John Koza

Koza mentioned that even though search plays a fundamental role in many AI approaches, he did not think that genetic methods were based on search. Several people disagreed, claiming that genetic methods relied mainly on search. After some discussion, it appeared that Koza based his comment on his belief that genetic algorithms and genetic programming were well defined, algorithmic procedures that had no non-deterministic components. But, of course, search methods are also well defined algorithmic procedures.

Smolensky argued that search methods were non-monotonic in the quality of solutions reached, meaning, presumably, that further search did not guarantee better solutions. In contrast with this view, Rumelhart, in the open discussion, claimed that all search is hill-climbing. Lesser asked if all of the other AI paradigms aren't simply search in disguise. Shoham asked for examples of AI problems that were not search problems. Korf replied that many problems of interest in AI have the characterstic that *reasonable* performance can be obtained in polynomial time, but that better performance requires search. And, because search requires time exponential in the branching factor, it was feasible only for problems whose branching factor was small or could be made effectively small by heuristic techniques.

Although not explicit in his abstract, Korf did distinguish two types of search processes, namely

1) search which incrementally *constructs* a solution (as in adding new cities to a traveling salesperson partial tour), and 2) search that *modifies* candidate solutions (as in changing the order of cities in a full tour).

2.1.2 Declarative Knowledge Bases and Logical Reasoning

Presentation by Hector Levesque

Abstract

One's approach to research in AI seems to depend to a large extent on what properties of intelligent behaviour one is most impressed by. For some, it might be the evolutionary antecedents of this behaviour in other animals; for others, its biological underpinnings in the central nervous system; still others, its societal dependencies. Those of us in knowledge representation and reasoning focus on what Zenon Pylyshyn has called the "cognitive penetrability" of intelligent behaviour: how our behaviour is conditioned in the amazingly flexible and general way that it is, by what we know, and especially by what we have been told about the world. This malleability through language is of course not evident at all levels of intelligent behaviour; it may be completely absent in aspects of motor control, early vision, speech, and so on. It is therefore quite possible to limit one's study to forms of behaviour where malleability of this sort is simply not an issue. But if we choose not to ignore this very real property of intelligent behaviour, the big question is: how can it possibly work.

Perhaps the most radical idea to emerge from AI (that is to say, the idea with the fewest historical antecedents) is in fact a proposed solution to this problem, due to John McCarthy: the vision of a system that would decide how to act in part by running formal reasoning procedures over a body of explicitly represented knowledge. The imagined system, called the Advice-Taker [McCarthy 58], would not so much be programmed for specific tasks as told what it needed to know, and expected to infer the rest somehow. Since this landmark 1958 paper, the idea of a system based on automated reasoning in this way has evolved into what is now known as knowledge-based systems which, in the form of so-called expert systems, still account for the vast majority of working AI programs.

Because the purpose of reasoning in such systems is to make explicit what is merely implied by what the system has been told, considerable attention has been devoted to understanding precisely the sort of implication involved, and its computational properties. In particular, the area of *knowledge representation and reasoning (KRR)* studies the computational task defined by: (1) a language, a declarative formalism for expressing knowledge; (2) a knowledge base, a collection of symbolic structures representing sentences from (1); and (3), an inference regime, a specification of what implications (assumptions, explanations, etc), meaningful according to the semantics of (1), should be extracted given (2) alone as input. Those interested primarily in (1), focus on languages and logics for representing knowledge, with special attention these days to defaults, probabilities, and definitions; those interested primarily in (2), focus on building knowledge bases representing aspects of the commonsense world, with a current emphasis on qualitative knowledge about the physical world; those interested primarily in (3), focus on reasoning procedures, with special attention to the computational tractability of the reasoning tasks involved. All three aspects are studied together and independently, and considerable progress has been made on all counts.

While the McCarthy idea was never expected to explain all of intelligent behaviour, it does appear to be the only proposal on the table with anywhere near the generality needed to account for its cognitive penetrability. Be that as it may, a wide variety of objections to this account of intelligent behaviour have been raised, and there is a call from many quarters for a return to more traditional approaches grounded in the neurosciences, robotics, and elsewhere. Unfortunately,

many of the objections raised are philosophical in nature (or based on deeply held feelings about what cannot be happening in our heads), and so are not refutable by any advances in our understanding of knowledge-based systems, or any computer systems, for that matter.

However, one major challenge to the knowledge-based system idea does appear to be purely computational in nature: it is, roughly, that

> The calculations required for this use of automated reasoning
> are too complex for even the most powerful machines.

In other words, the objection is that while perhaps formal reasoning procedures can give us some insight into what is happening when people engage in logic problems, puzzle-solving, or preparing tax returns, the computational tasks involved are far too demanding to account for "ordinary thinking," which necessarily involves real-time interaction with a very dynamic environment. Logical reasoning of the kind contemplated by McCarthy, in particular, is simply too hard to be biologically plausible.

But is it? For all I know, it might be, and I certainly will not attempt to refute the objection here. On the other hand, there is a growing body of research on aspect (3) above that suggests that provided we do not hold too strongly to mathematical elegance and parsimony, automated reasoning can be broadly useful, semantically coherent, and yet still computationally manageable. By themselves, these results do not imply anything like biological plausibility; but they do suggest that claims about implausibility need to be made more carefully.

Discussion led by Tom Mitchell

Mitchell began the discussion by asking whether the KRR people themselves ought to be so concerned with the problem of intractability when so many people in AI are already working with tractable methods. Rich Korf added that, in any case, the intractability results are "worst case," and ordinary problems are typically not worst case. Vic Lesser noted that "satisficing" is one way to achieve tractability and wondered whether or not satisficing could be formalized in a logical manner. Pattie Maes mentioned a seeming paradox, namely that the computational complexity of logical reasoning increases as the knowledge base grows in size, whereas it seems that the more knowledge a person has, the more readily that person can arrive at a useful conclusion.

In Levesque's talk, he mentioned that tractability can sometimes be achieved by allowing unsound (but nevertheless usually useful) reasoning. He and colleagues are working on an interesting technique for unsound deduction. Their system, called GSAT [Selman 92], first finds a number of models, $M_1, \ldots, M_n$ that satisfy a knowledge base. Then, GSAT concludes (heuristically and unsoundly) that an arbitrary formula α follows from the knowledge base just in case all n of the M_i satisfy α. GSAT uses a greedy search to find models that satisfy the knowledge base. The system can learn from its mistakes (that is, when it incorrectly concludes that some β follows from the knowledge base) by adjusting its models.

2.1.3 The Blackboard Architecture as a Foundation for Adaptive Intelligent Systems

Presentation by Barbara Hayes-Roth

Abstract

I discuss the blackboard architecture [Hayes-Roth 85] as a foundation for adaptive intelligent systems—systems that perceive, reason about, and interact with other dynamic entities in real time. Examples are intelligent mobile robots and monitoring agents.

The standard blackboard architecture has three key elements. The "blackboard" is a global data structure, frequently organized into levels of abstraction and orthogonal spatial or temporal dimensions, in which reasoning results are recorded and organized. Independent "knowledge sources" embody diverse reasoning actions that are reactively enabled by environmental events. The "scheduler" is a procedure that iteratively identifies executable instances of these actions and chooses the "best" one to execute next, based on general and domain-dependent criteria. Strengths of the architecture are: integration of diverse knowledge and reasoning actions; coordination of reasoning at multiple levels of abstraction; knowledge-level modifiability; and opportunistic control of reactive reasoning.

My own work on the blackboard architecture introduced "dynamic control planning," in which an agent dynamically constructs plans for its own behavior at run time. Like any other reasoning, control planning is performed by reactively enabled knowledge sources that modify the contents of the blackboard under the control of the scheduler. The scheduler itself is greatly simplified. Instead of fixed scheduling criteria, it uses whatever control plans are active at the current point in time. Because control plans can be abstract, they typically do not determine a unique sequence of actions, but only constrain the set of acceptable sequences. As a consequence, the agent's actual behavior emerges from the interaction between its current control plan and the actions that happen to be enabled by environmental events. Control plans also provide a data base for explaining behavior and a language for accepting advice and for recognizing and reusing other agents' plans.

More recent extensions to the architecture exploit control plans in other important ways. First, perception subsystems use control plans to determine what is the most important information to perceive in a complex environment. Second, a global planner coordinates task-specific control plans for loosely-coupled tasks, such as fault detection, diagnosis, prediction, and planning. Third, a resource-limited scheduler uses control plans to restrict its consideration of alternative actions, identifying the best one to execute next in short, constant time, despite substantial increases in event rate and number of known actions.

The extended blackboard architecture has several significant weaknesses. It is underconstraining in the sense that it allows a range of application development styles, some of which may not exploit the architecture's strengths. Its operation is fundamentally sequential, but would benefit from fine-grained parallelism, especially in the identification of executable actions and other memory search functions. It is incomplete and does not address several critical cognitive capabilities, such as: episodic memory organization, dynamic memory processes, involuntary learning, and natural language.

Discussion led by Pattie Maes

Maes began by claiming that the blackboard architecture should be viewed as a "methodology" or set of principles. There are no theorems or algorithms that come along with this methodology. The programmer is obliged to add much domain-specific knowledge. The blackboard methodology can be contrasted with that of some other search-based, domain-independent methods, which can

be applied directly to many problems. Thus, the blackboard architecture (and Soar too, for that matter) should be viewed more as programming languages than as problem-solving architectures, and they should be evaluated as such.

2.1.4 Soar

Presentation by Paul Rosenbloom (given in his absence by John Laird)

Abstract

The Soar project is a long-term, distributed, multi-disciplinary, evolutionary effort to construct, understand, and apply an architecture that supports the construction of both an integrated intelligent system and a unified theory of human cognition [Rosenbloom 93]. Active Soar research topics include intelligent agents in simulation environments; integrated frameworks for learning and planning; learning from advice and experimentation; natural language comprehension, generation, and acquisition; computationally-bounded and neural implementations; interaction with external tools; expert systems (in medicine, music, etc.); tutoring; and models of human behavior in such domains as typing, browsing, interaction, syllogisms, and garden path sentences.

This talk focuses on the central cognitive aspects of Soar, leaving to John Laird's talk the task of situating Soar within the external world. The approach is to characterize Soar via a hierarchy of four computational levels, while exemplifying its behavior at each level via recent work on planning and learning. Starting from the top, the four levels are: the Knowledge Level, the Problem-Space Level, the Symbol Level, and the Implementation Level.

At the Knowledge Level an intelligent agent comprises a set of goals, a physical body, a body of knowledge, and decision making via the Principle of Rationality (that the agent will intend actions that its knowledge says will achieve its goals). This level is useful here primarily as an idealized specification of the behavior Soar is to exhibit, and thus as a justification for the content of the lower levels. However it is an idealization that is not usually achievable in the real world. Learning at the Knowledge Level corresponds to the acquisition of new knowledge. Planning disappears as a distinct process at this level, lost in the abstraction provided by the Principle of Rationality.

At the Problem-Space Level an intelligent agent comprises a set of interacting problem spaces, each consisting of a set of states and a set of operators. Behavior is driven by a set of four primitive components that jointly determine how the Problem-Space Level implements the Knowledge Level. These components are themselves recursively defined as intelligent systems (i.e., as a hierarchy of levels topped off by the Knowledge Level). The recursion terminates with "available knowledge", that is knowledge that can be brought to bear directly without descending from the Knowledge Level to the Problem-Space Level.

The Problem-Space Level is undergirded by a set of principles about the construction of intelligent systems, such as: reasoning occurs in constrained micro-worlds, action is separated from control, control is based on all available knowledge, search arises because of uncertainty (lack of knowledge), and behavior is represented uniformly from knowledge lean to strong. Planning at the Problem-Space Level is the recursive use of problem spaces because of a lack of available knowledge (i.e., plans). It can take the form of a wide range of planning methods, as well as combinations of methods. Learning at the Problem-Space Level is the addition of new available knowledge. Such learning has a significant impact on the planning process (and on the plans themselves), and vice-versa.

At the Symbol Level an intelligent agent is a collection of memories and processes. The Soar architecture is one instantiation of the Symbol Level that has been constructed to support the Problem-Space Level (and through it, the Knowledge Level). At its core are a set of five

distinct memories, plus six processes that operate on these memories. Control is mediated by a behavioral cycle that intermingles perception, action, memory access, decision making, and learning. Planning at this level is like planning at the Problem-Space Level, but with the further refinement of a modular, highly conditional and situation-sensitive plan representation, based on sets of associations. Learning is the creation of new associations from behavioral traces.

At the Implementation Level the key issues are technological ones of speed, boundedness, etc. The most recent version of Soar – Soar6 – is a complete reimplementation in C. It is 10-40 times faster than Soar5, and is competitive with the fastest AI architectures, requiring 4 msec to fire productions and 100 msec to make decisions, even with very large memories of over 100,000 rules. Progress has been made on boundedness, but there is much more still to be done.

Discussion led by Barbara Hayes-Roth

Hayes-Roth began by observing that some properties of Soar are difficult to understand, for example it would be hard to predict what Soar would do in various circumstances. Richard Korf wondered how Soar would fare when faced with a combinatorial explosion. Laird conceded that Soar would probably not be appropriate in circumstances requiring exponential searches that are unguided by control knowledge. Nils Nilsson observed that the layered approach to describing Soar was appealing because one could pick and choose from among the layers one wanted to endorse.

2.2 Session 2: Biocomputational Approaches

2.2.1 Overview by David Rumelhart

To set a context for this session, Rumelhart contrasted the symbol-system and biocomputational approaches. The former is motivated by attempts to model the high level reasoning abilities of humans whereas the latter is inspired more by simpler aspects of intelligence. In the biocomputational approach, one also is motivated by evolutionary considerations, asking how did this level of behavior arise. Biocomputationalists begin with a concern for sensorimotor abilities and adaptation. It is worth pointing out that most of the cerebral cortex in primates is devoted to sensorimoter activities, and the rest of the cortex appears to have the same structure as the sensorimotor areas. A reasonable hypothesis then is that even abstract intelligent behavior uses the methods of the sensorimotor system. Therefore the biocomputational approach may be the best route even to high level AI.

2.2.2 Second-Wave Connectionism

Presentation by Paul Smolensky

Abstract

Introduction

Connectionism can be viewed as the study of the following hypothesis:

- Intelligence arises from the interaction of a large number of simple numerical computing units—abstract neurons—whose interconnections contain knowledge which is extracted from experience by learning rules.

'First-wave connectionism' presumed that a consequence of this hypothesis is that cognition is association, free of all symbols and rules, that cognition is pervasively graded or 'soft', and that all domain knowledge is statistically extracted from experience. 'Second-wave connectionism' regards this as a mistake, that the hypothesis is fully compatible with formal descriptions of cognition as the manipulation of complex symbolic structures via rules or subject to constraints, where certain aspects of cognition are discrete and 'hard', and where some knowledge arises from sources other than statistical induction.

Second-Wave Connectionism develops a uniform computational framework for both the 'soft' and the 'hard' facets of cognition by combining methods of continuous and discrete mathematics. The long-term goal of Second-Wave Connectionism is the development of cognitive models and AI systems employing radically new techniques for representation, processing and learning which combine continuous/'connectionist' and discrete/'symbolic' elements. Such radical techniques lie in the future, but more conservative means of unifying connectionist and symbolic formalisms have already provided significant advances in the study of Universal Grammar, as summarized below.

Some Principles of Second-Wave Connectionism

- Information is represented in widely distributed activity patterns—activity vectors—which possess global structure describable through discrete data structures.

The most general technique here is 'tensor product representations'. These are distributed activation vectors S possessing the following global structure:

$$S = \sum_i R_i * F_i$$

where capital letters denote vectors and '$*$' denotes the tensor (generalized outer) product operation on vectors. This structured vector S is analyzed as a symbolic structure s defined by a set of role/filler bindings $s = \{r_i/f_i\}$. The analysis can be done recursively to treat embedding; if an embedded filler f_i is itself a complex structure, then it corresponds recursively to a set of bindings of sub-roles and sub-fillers, $f_i = \{sr_{i,j}/sf_{i,j}\}$, and

$$F_i = \sum_j SR_{i,j} * SF_{i,j}$$

so that the original structure s corresponds to

$$S = \sum_i \sum_j R_i * SR_{i,j} * SF_{i,j}$$

Tensor calculus allows arbitrarily many tensor products and thereby arbitrarily deep embedding. Recursively defined roles r_i such as the positions in a binary tree correspond to recursively defined vectors R_i, enabling distributed activity patterns to possess global recursive structure.

- Information is processed in widely distributed connection patterns—weight matrices—which possess global structure describable through symbolic expressions for recursive functions.

The global structure of the distributed activity vectors given by tensor product representations is structure that can be effectively processed in connectionist networks, giving rise to massively parallel structure processing. Complex structure-manipulations can be done in one-step connectionist operations, implemented as multiplication by weight matrices with appropriate global structure. These operations can compute recursive functions, when the activity vectors being processed have recursive structure, and when the weight matrices meet a simple structural

condition. The global structure of weight matrices for computing particular recursive functions can be precisely characterized using a simple (symbolic) programming language.

These methods have been applied to the study of grammars of natural and formal languages, based on two motivations. The first is that it has generally been assumed that the prospects for connectionism contributing to the theory of grammar are grim, given what is generally taken to be the *terrible* theoretical fit between connectionism and grammar. The second motivation is the conviction that, in fact, there is an *excellent* fit between connectionism and grammar, as revealed by the following principle:

- Information processing constructs an output for which the pair (input, output) is optimal in the sense of maximizing a connectionist well-formedness measure called Harmony (or minus 'energy'). The Harmony function encapsulates the system's knowledge as a set of conflicting soft constraints of varying strengths—the output constitutes the optimal degree of simultaneously satisfying these constraints.
- A grammar (regarded as a function mapping an input string to an output = its parse) operating on this principle is a Harmonic Grammar.

Harmonic Grammar

The component of the above principle concerning grammar can be elaborated:

- Grammars are sets of 'rules' which govern complex, richly structured symbolic expressions. These 'rules' are actually constraints which are:
  - parallel/simultaneous
  - soft/violable
  - conflicting
  - of differing strengths.

  Each constraint assesses an aspect of well-formedness/Harmony. The grammar assigns the optimal/maximal-Harmony parse as the output best satisfying the constraint set.

This principle has been extended as the basis of the following conception of Universal Grammar:

- Universal Grammar specifies a set of constraints holding in all languages. Cross-linguistic variation arises primarily as different languages assign different relative strengths to the universal constraints, giving rise to different patterns of resolving conflicts between constraints.

This principle constitutes a very significant innovation in the theory of natural language grammar, arising directly from principles of Second-Wave Connectionism.

Harmonic Grammar has been successively applied to a detailed account of the complex interactions between conflicting syntactic and semantic constraints in a set of grammaticality judgements which could not be accounted for using traditional methods.

Extensive studies of phonology soon to be circulated in a book-length manuscript constitute a further development of Harmonic Grammar showing that, arguably for the first time, a theory of Universal Phonology is made possible by this formal treatment of systems of conflicting soft constraints. Crucially, these studies show that in many areas, universal phonology calls for a notion of 'different strengths' of soft constraints which is *non-numerical*: each constraint has absolute priority over all weaker constraints. Thus each language ranks the constraints of Universal Phonology in its own strict dominance hierarchy, and strikingly different phonological systems emerge as the same set of Universal well-formedness constraints interact through different domination hierarchies. This framework, Optimality Theory, is joint research of Alan Prince and myself.

Given its success with natural language grammar, it is natural to ask if Harmonic Grammar has the expressive power to specify formal languages. The exact form of the constraints in Harmonic Grammar arises from simple underlying connectionist processing mechanisms, and is thus very simple and seemingly very restrictive. However it can be shown that formal languages at all levels in the Chomsky hierarchy can be specified by Harmonic Grammars.

Discussion led by Nils Nilsson

Nilsson asked about the possibility of hybrid systems. Interesting as it might be from the point of view of neuroscience that list-processing operations can be performed by connectionist networks, engineers are already quite happy with their methods for list processing. As one ascends from the lower level mental functions (such as pattern recognition) that are well handled by connectionist networks, won't there be a point at which the engineer (if not the neuroscientist) might want to switch to more standard symbol-manipulation methods? Smolensky stressed that the (second-wave) connectionist implementations conferred added advantages mentioned in his presentation, namely that they permitted parallel computations and that they could deal with soft or violable or even conflicting constraints of differing strengths.

2.2.3 Genetic Programming

Presentation by John R. Koza

Abstract

Since the invention of the genetic algorithm by John Holland in the 1970's [Holland 75], the genetic algorithm has proven successful at searching nonlinear multidimensional spaces in order to solve, or approximately solve, a wide variety of problems. The genetic algorithm has been particularly successful in solving optimization problems where points in the search space can be represented as chromosome strings (i.e., linear strings of bits). However, for many problems the most natural representation for the solution to the problem is a computer program or function (i.e., a composition of primitive functions and terminals in tree form), not merely a single point in a multidimensional search space. Moreover, the size and shape of the solution to the problem is often unknowable in advance.

Genetic programming [Koza 92] provides a way to genetically breed a computer program to solve a surprising variety of problems. Specifically, genetic programming has been successfully applied to problems such as:

- evolution of a subsumption architecture for controlling a robot to follow walls or move boxes,
- discovering control strategies for backing up a tractor-trailer truck, centering a cart, and balancing a broom on a moving cart,
- discovering inverse kinematic equations to control the movement of a robot arm to a designated target point,
- emergent behavior (e.g., discovering a computer program which, when executed by all the ants in an ant colony, enables the ants to locate food, pick it up, carry it to the nest, and drop pheromones along the way so as to recruit other ants into cooperative behavior),
- symbolic "data-to-function" regression, symbolic integration, symbolic differentiation, and symbolic solution to general functional equations (including differential equations with initial conditions, and integral equations),

- Boolean function learning (e.g., learning the Boolean 11-multiplexer function and 11-parity functions),
- classification and pattern recognition (e.g., distinguishing two intertwined spirals),
- generation of high entropy sequences of random numbers,
- induction of decision trees for classification,
- optimization problems (e.g., finding an optimal food foraging strategy for a lizard),
- sequence induction (e.g., inducing a recursive computational procedure for generating sequences such as the Fibonacci sequence), and
- finding minimax strategies for games (e.g., differential pursuer-evader games, discrete games in extensive form) by both evolution and co-evolution.

In genetic programming, the individuals in the population are compositions of primitive functions and terminals appropriate to the particular problem domain. The set of primitive functions used typically includes arithmetic operations, mathematical functions, conditional logical operations, and domain-specific functions. The set of terminals used typically includes inputs appropriate to the problem domain and various constants.

The compositions of primitive functions and terminals described above correspond directly to the computer programs found in programming languages such as LISP (where they are called symbolic expressions or S-expressions). In fact, these compositions correspond directly to the parse tree that is internally created by the compilers of most programming languages. Thus, genetic programming views the search for a solution to a problem as a search in the hyperspace of all possible compositions of functions that can be recursively composed of the available primitive functions and terminals.

Genetic programming, like the conventional genetic algorithm, is a domain independent method whose execution consists of three steps. Genetic programming proceeds by genetically breeding populations of compositions of the primitive functions and terminals (i.e., computer programs) to solve problems by executing the following three steps:

1. Generate an initial population of random computer programs composed of the primitive functions and terminals of the problem.
2. Iteratively perform the following sub-steps until the termination criterion for the run has been satisfied:
   (a) Execute each program in the population so that a fitness measure indicating how well the program solves the problem can be computed for the program.
   (b) Create a new population of programs by selecting program(s) in the population with a probability based on fitness (i.e., the fitter the program, the more likely it is to be selected) and then applying the following primary operations:
      i. Reproduction: Copy an existing program to the new population.
      ii. Crossover: Create two new offspring programs for the new population by genetically recombining randomly chosen parts of two existing programs.
3. The single best computer program in the population produced during the run is designated as the result of the run of genetic programming. This result may be a solution (or approximate solution) to the problem.

The genetic crossover (sexual recombination) operation operates on two parental computer programs and produces two offspring programs using parts of each parent. Specifically, the crossover operation creates new offspring by exchanging sub-trees (i.e., sub- lists, subroutines, subprocedures) between the two parents. Because entire sub-trees are swapped, this crossover operation always produces syntactically and semantically valid programs as offspring regardless of the choice of the two crossover points. Because programs are selected to participate in the

crossover operation with a probability proportional to fitness, crossover allocates future trials to parts of the search space whose programs contains parts from promising programs.

Automatic function definition automatically and dynamically enables genetic programming to define potentially useful functions dynamically during a run.

Discussion led by Andrew Barto

Barto asked about the "economics" of genetic programming. How efficient is it compared, say, to other learning methods or to hand-crafting programs? There was also discussion about combining learning methods and genetic methods. Such combinations would permit Lamarkian evolution. Previous work done by Stork has shown an apparent influence of learning on evolution known as the Baldwin effect. Someone asked about whether or not the fitness function could include a factor that would penalize systems for large programs. John Koza felt that favoring parsimony might be counterproductive. In order to probe the limits of genetic programming, Rich Korf asked for examples of where it had failed to work. Koza didn't provide any such examples.

2.2.4 Behavior-based Artificial Intelligence

Presentation by Pattie Maes

Abstract

Introduction

Since 1985, a new wave has emerged in the study of Artificial Intelligence (AI). At the same moment at which the popular, general belief is that AI has been a "failure", many insiders believe that something exciting is happening, that new life is being brought to the field. The new wave has been termed "behavior-based AI" as opposed to main-stream "knowledge-based AI", or also "bottom-up AI" versus "top-down AI". Behavior-based AI poses problems in a different way, investigates interesting new techniques and applies a set of different criteria for success. Behavior-based AI is not limited to the study of robots, but rather presents itself as a general approach for building autonomous systems that have to deal with multiple, changing goals in a dynamic, unpredictable environment. This includes applications such as interface agents [Maes 93a], process scheduling [Malone 88], and so on.

Problem Studied

The goal of both knowledge-based AI and behavior-based AI is to synthesize computational forms of intelligent systems. Both approaches attempt to model intelligent phenomena such as goal-directed behavior, prediction, learning, communication and cooperation. Knowledge-based AI has traditionally emphasized the modeling and building of systems that "know" about some problem domain. These systems model the domain and can answer questions about this problem domain, often involving extensive problem solving and reasoning. Behavior-based AI on the other hand has emphasized the modeling and building of systems that "behave" in some problem domain.

Solutions Adopted

The difference between knowledge-based AI and behavior-based AI lies not only in the problems that are studied, but also in the techniques and solutions that are explored. The solutions adopted in main-stream AI projects can be characterized as follows:

- modular decomposition along functional modules such as preception, knowledge representation, planning, etc
- these functional modules are general, domain-independent

- the emphasis is on an internal representation of the world
- the control architecture is sequential
- activity is viewed as the result of deliberative thinking
- learning mostly consists of compilation or reformulation of knowledge

In contrast, behavior-based AI adopts the following solutions:

- modular decomposition along task-achieving modules
- these modules are highly task-dependent
- there is no central, declarative representation
- the architecture is highly distributed
- activity is an emergent property of the interactions between the modules and the environment
- learning from experience (and evolution) are emphasized

Insights
The methods developed by behavior-based AI are grounded in two important insights:

- Looking at complete systems changes the problems often in a favorable way.
- Interaction dynamics can lead to emergent complexity.

All of these points are elaborated upon in [Maes 93b].

Discussion led by Mike Wellman

Wellman noted that knowledge-based AI systems were not as robust nor did they degrade as gracefully as we would like. But might these defects be due simply to the fact that these qualities were fundamentally difficult to achieve regardless of the method? Aren't the behavior-based methods simply off-loading these difficult problems onto the programmer? Pattie Maes said that the programmer of behavior-based systems does, in fact, have to do more work, but that is to be expected due to the nature of the problem. Tom Mitchell made the interesting observation that rule-based expert systems are somewhat similar in spirit to behavior-based methods, because the expert systems use phenomenological, often ad hoc, rules rather than deep (knowledge-based) models. It was also observed that the behavior-based methods, so far, have been applied to quite different problems than those attacked by the knowledge-based methods.

Another interesting description of this approach to AI is contained in a paper by Stuart Wilson who calls it the "animat" approach. [Wilson 91].

2.2.5 Real-time Learning and Control

Presentation by Andrew G. Barto

Abstract

As AI researchers become more concerned with systems embedded in environments demanding real-time performance, the gulf narrows between problem-solving/planning research in AI and control engineering. Similarly, machine learning methods suited to embedded systems are comparable to methods for the adaptive control of dynamical systems. Although much of the

research on learning in both symbolic and connectionist AI continues to focus on supervised learning, or learning from examples, there is increasing interest in the *reinforcement learning* paradigm because it addresses problems faced by autonomous agents as they learn to improve skills while interacting with dynamic environments that do not contain explicit teachers.

I describe the contributions of a number of researchers who are treating reinforcement learning as a collection of methods for successively approximating solutions to stochastic optimal control problems. See also [Barto 91]. Within this framework, methods for learning heuristic evaluation functions by "backing up" evaluations can be understood in terms of Dynamic Programming (DP) solutions to optimal control problems. Such methods include one used by Samuel in his checkers playing program of the late 1950s, Holland's Bucket-Brigade algorithm, connectionist Adaptive Critic methods, and Korf's Learning-Real-Time-A* algorithm. Establishing the connection between evaluation function learning and the extensive theory of optimal control and DP produces a number of immediate results as well as a sound theoretical basis for future research.

As applied to optimal control, DP systematically caches into permanent data structures the results of repeated shallow lookahead searches. However, because conventional DP requires exhaustive expansion of all states, it cannot be applied to problems of interest in AI that have very large state sets. DP-based reinforcement learning methods approximate DP in a way that avoids this complexity and may actually scale better to very large problems than other methods.

Discussion led by Richard Korf

Korf raised the issue that most of the successful examples of control-theory applications required storing in memory every state of the problem space that was explored, and that this would be infeasible for combinatorial spaces. In that case, something like Samuel's evaluation learning technique for checkers was required to generalize over large parts of the search space. While in principle similar approaches could be applied here, none of the theoretical results would then apply, and even experimental progress would become much more difficult. Barto agreed that this was a very important research direction.

2.3 Session 3: Heterogeneous Approaches

2.3.1 Distributed Artificial Intelligence

Presentation by Victor Lesser

Abstract

As more AI applications are being formulated in terms of spatial, functional or temporal distribution of processing, Distributed Artificial Intelligence (DAI) is emerging as an important subdiscipline of AI. This is especially true as the outlines for computing in the next century are beginning to emerge: networks of cooperating, intelligent heterogeneous agents in which agents can be both man and machine. The need to cooperate in such systems occurs not only from contention for shared resources but also because agents are solving sets of interdependent subproblems. Achieving consistent solutions to interdependent subproblems can be difficult due to the infeasibility of each agent having readily accessible an up-to-date, complete and consistent (with respect to other agents) view of the information necessary to solve its subproblems completely and accurately. This lack of appropriate information arises from a number of factors:

- Limited communication bandwidth and the software costs of packaging and assimilating information communicated to agents,

- The dynamically changing environment due not only to hardware and software failures but also to agents entering and exiting the computational network,
- The heterogeneity of agents which makes it difficult to share information,
- The potential for competitive agents who, for their own self-interest, are not willing to share certain information.

Often in describing DAI, the field is split into two research areas: cooperative, distributed problem solving where the goal of cooperation is programmed into the agent architecture, and multi-agent systems in which cooperation comes out of self-interest. However, this distinction may not be substantive since at the heart of both sub-areas is the need to deal with uncertainty caused by the lack of adequate information.

In order to deal with this uncertainty, there have been a number of specific techniques both formal and heuristic that have been developed by DAI researchers. The general principles guiding the development of these techniques are the following:

The system design goal of producing the optimal answer with minimal use of communication and processing resources while at the same time being able to respond gracefully to a dynamically changing environment is often unrealistic for most real-world DAI tasks. Instead, a "satisficing" criterion for successful system performance is often adopted based on achieving, over a wide range of environmental scenarios, an "acceptable" answer using a "reasonable" amount of processing resources.

The resolution of uncertainty (inconsistent, incomplete and incorrect information) should be an integral part of network problem solving rather than something built on top of the basic problem-solving framework. This process of resolution is in general a multi-step, incremental process (sometimes thought of as negotiation) involving a cooperative dialogue among agents. Further, resolution of all uncertainty may not be necessary for meeting the criteria of acceptable system performance.

Sophisticated local control is in many cases necessary for effective network problem solving. Agents need to explicitly reason about the intermediate states of their computation (in terms of what actions they expect to take in the near term, what information from other agents would be valuable for making further progress in their local problem solving, etc.), and to exploit as best as possible the available information. Agents also need to be able to acquire, represent and reason about beliefs concerning the state of other agents, and to use assumptions about the rationality of other agents' problem solving in their reasoning.

Organizing the agents in terms of roles and responsibilities can significantly decrease the computational burden on coordinating their activities. However, these assignments should not be so strict that an agent does not have sufficient latitude to respond to unexpected circumstances, nor should they be necessarily fixed for the duration of problem solving. Organizational control should be thought of as modulating local control rather than dictating.

DAI is still in its infancy, and there is only a relatively small group of active researchers. I expect significant intellectual strides to occur in the near term as the strength or weaknesses of current ideas are evaluated in real applications and more researchers get involved.

Discussion led by John Laird

Laird asked if it might be possible to describe at the knowledge level a group of knowledge-level agents. Such a description would be needed if we wanted to describe an organization, for example, at the knowledge level. This question in turn entails the question of what new knowledge must be added to knowledge-level agents in order to make the organization work effectively. And, would this additional knowledge require a new kind of architecture for knowledge level agents as components of an organization? On a different topic, Richard Korf wondered whether various AI

systems exploiting parallel computation, for example parallel search operations, were examples of DAI systems.

2.3.2 Economic Approaches to Artificial Intelligence

Presentation by Michael P. Wellman

Abstract

Economics is the study of allocating scarce resources by and among a distributed set of individuals. The best-developed economic theories are those that take the individuals to be rational agents acting in their own self-interest. To take an economic perspective on Artificial Intelligence is to focus on the decision-making aspects of computation: that the result of deliberation is to take action allocating resources in the world. At the level of the individual, the natural economics-oriented goal is to design "rational" decision-making agents. At the level of a collection of agents, the goal is to design a decentralized mechanism whereby rational self-interested agents produce socially desirable aggregate behaviors.

The most pervasive economic characterization of rationality is that of (Bayesian) decision theory [Savage 72]. Decision theory generalizes goal-based notions of rationality to admit graded preferences for outcomes and actions with uncertain effects. However, since decision theory *per se* addresses only behavior and not process, a fully computational account of how to design rational agents is not immediately available. Incorporating decision-theoretic concepts into Planning (the decision-oriented sub-discipline of AI) presents special technical challenges that we have only begun to address [Wellman 93a]. However, recent advances in representing probabilistic knowledge (e.g., Bayesian or probabilistic networks [Pearl 88], first-order probabilistic logics [Halpern 90]), and concern with planning under uncertainty has led to increasing interest in probabilistic methods. Work on representing preferences is also emerging, with special attention to techniques reconciling utility-theoretic and goal-oriented approaches [Wellman 91]. Although progress on the general problem is quite preliminary, work on specific applications suggests that even restrictive methods accounting for uncertainty and partial satisfiability of objectives can extend the scope of AI planning systems.

Most of economic science is devoted to the study of decision making by multiple distributed agents. The class of mechanisms studied most deeply by economists is that of market price systems. In a market price system, agents interact only by offering to buy or sell quantities of commodities at fixed unit prices. Agents may simply exchange goods (consumers), or may also transform some goods into other goods (producers). In a computational version of a market price system, we can implement consumer and producer agents and direct them to bid so as to maximize utility or profits, subject to budget or technology constraints. Under certain technical assumptions, the equilibria of this system correspond to desirable or optimal resource allocations. In such cases, the artificial economy serves as an effective distributed computing system for deriving socially desirable global activities. I have implemented a "market-oriented programming" environment for specifying computational economies, with bidding protocols and mechanisms for finding competitive equilibria. Initial experiments with multicommodity flow problems have demonstrated basic feasibility of the approach; work is ongoing on extensions and other applications [Wellman 93b].

Discussion led by Dave Waltz

[No record of discussion of this talk]

2.3.3 Massively Parallel Artificial Intelligence

Presentation by Dave Waltz

Abstract

Advances in hardware and computer architecture continue to change the economics of various AI (as well as all other) computing paradigms. Workstation chips are the main driving force: they provide by far the greatest computational power per dollar, and are quickly causing the demise of minicomputers and mainframes, and the blurring of the line with PCs. The new generation of massively parallel machines—built out of large numbers of workstations chips, and designed as servers for desktop workstation clients—extends the potential for applications at the high end of the computing spectrum, offering higher computing and I/O performance, much larger memories, and MIMD as well as SIMD capabilities. Computing costs for the same level of performance have dropped by about 50% per year for the last few years, and will continue to drop steeply for the foreseeable future.

The trends noted have clear consequences for AI: most applications, research, and development will be done on workstations, and the most computationally intensive AI tasks will migrate to massively parallel machines. What kinds of AI tasks are these? Only those that will not be doable on workstations in the next five to ten years. These high end tasks fall into three main categories: 1) database-related applications and research; 2) applications that combine AI with science and engineering; and 3) "truly intelligent systems," covering very large scale neural and/or cognitive models. Very large databases offer the most promising areas for near and medium term AI efforts involving massively parallel processors. For example, much larger knowledge bases can be stored and accessed quickly, even if complex inferences must be made; learning methods and simple-to-program brute-force methods for decision support can replace hand coding, allowing much more cost-effective system-building; and just about any parallel AI paradigm should be capable of executing efficiently and quickly.

A number of prototype and fielded systems have been built and evaluated on the massively parallel Connection Machine CM-5 over the last few years. The projects include: automatic keyword assignment for news articles using MBR nearest-neighbor methods (MBR = Memory-Based Reasoning); automatic classification of Census Bureau long forms by occupation and industry of the respondent; time series prediction for financial data and for chaotic physical systems, using artificial neural nets and MBR; protein structure prediction using MBR together with backpropagation nets, and statistics; work on "database mining" using a variety of methods, including genetically-inspired algorithms, MBR, ID-3, and others; and the generation of graphics using genetically-inspired operations on s-expressions. (See [Kitano 91].)

One of the other benefits of having large amounts of computing power is that many different methods (and many variants on each) can be tried, since each takes only a short time to run. In the course of our work, we have been able to compare a large number of different algorithms, and this in turn has helped us to see these different methods as a set of engineering choices, each with its own strengths and weaknesses, and best areas of application. In some cases we have found that methods are complementary, and have built hybrid systems that combine the results of several methods, yielding performance that is superior to any single method.

Discussion led by David Rumelhart

Rumelhart observed the similarity of Waltz's general approach to that of nearest-neighbor methods, and this observation led to a discussion of the differences between neural nets and nearest-neighbor methods in pattern recognition. Each of the two has given better results than the other on certain applications.

2.3.4 Agent-Oriented Programming

Presentation by Yoav Shoham

Abstract

Is there a clear, general, and nonvacuous theory of agenthood in AI?

The short answer, in my view, is no, but we might be inching towards one. Among the ingredients of such a theory (and this is already lengthier, and anything but crisp) might be the following:

- Agents function in an environment; the analysis and evaluation is always of an agent-environment pair (analogy to the kinematic pair in kinematics),
- The agents interact with the environment autonomously through "sensors and effectors"; it is unclear at this time whether this can be distinguished from ordinary I/O,
- The environment contains other agents; agents "interact" with one another,
- Agents have continuous existence.

All of the above are pretty much uncontroversial; the following are less so:

- The environment is noisy and unpredictable,
- Agents communicate with one another,
- Agents possess explicit symbolic information, including information about one another,
- Agents have mental state,
- Agents function in 'real time',
- Agents belong to a social system.

Despite the unclarity of the concept, I believe that the notion of software agents will be a useful one. I sense informed interest on the part of software developers, and see an opportunity to pursue interesting basic research questions.

The micro-view: agents that have mental state

Formal theories of mental state have been around in AI for quite a while. For the most part, the motivation has been that of analysis. For example, Question-Answering systems have tried to model the user's beliefs and goals in order to provide informative answers. More recently, Agent-Oriented Programming has proposed using these theories for design, rather than mere analysis. In particular, in AOP [Shoham 93a] one programs software agents as if they have a mental state, including (for example) beliefs about one another.

Among the mental attitudes studied in AI are belief, desire, intention (BDI agent architectures), goal, commitment, decision, obligation.

By far the best studied and understood are the categories of knowledge and belief; the others lag behind significantly.

Characteristics of formal theories of mental attitudes in AI:

- They are crude in comparison with their natural counterparts,
- Despite the crudeness, they are useful in circumscribed applications,
- Despite the crudeness, they sometime shed light on the natural concepts.

The macro-view: dynamic evolution of conventions among agents

The presence of multiple agents calls for some form of coordination among them. Certain 'social laws' can be imposed at design time [Shoham 93b]; others emerge dynamically at run time [Shoham 92]. A number of disciplines have modeled phenomena in their domain in terms of complex dynamical systems, and have studied these systems both analytically and experimentally. These disciplines include:

- Mathematical sociology
- Mathematical economics
- Population genetics
- Statistical mechanics
- Control theory

The class of properties usually studied includes various convergence properties, as a result of local interactions in the domain. However, although superficially similar, the details of these systems are sufficiently different that it appears to be quite hard to transfer more than suggestive terminology between the different frameworks. (Fortunately or unfortunately, researchers in each of these communities have usually been insufficiently aware of the other discipline to be bothered by this fact.)

In studying the organization of multi-agent systems, and in particular ways in which to encourage coordination among the various agents, AI faces the problem of designing local interactions in ways which lead to attractive global properties. There appears to be an important difference from the frameworks mentioned above, in that the transitions in these other dynamical systems depend in part on global properties of the systems (e.g., fitness function in population genetocs, distribution of strategies in mathematical economics). In contrast, in multi-agent systems it is often unreasonable to assume the availability of such global knowledge (control theory may share this purely local flavor; this remains to be investigated). Initial computer simulations reveals some surprising system dynamics, but this is largely virgin territory which awaits exploration.

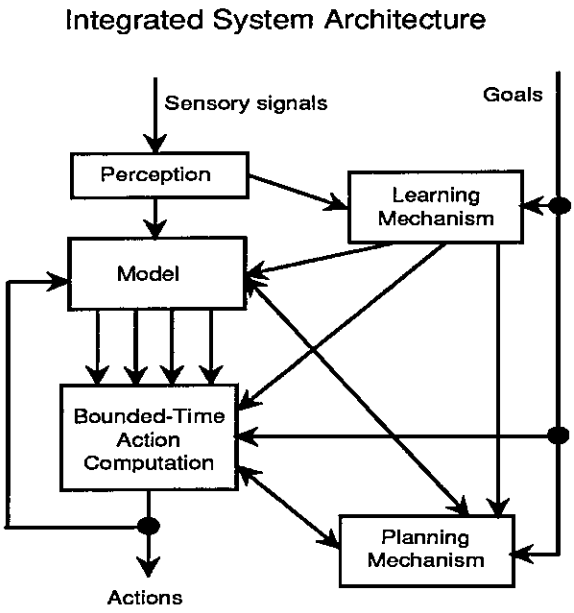
Discussion led by Victor Lesser

Lesser questioned whether or not the AOP approach would be competitive with other DAI methods. He also wondered about the ability of AOP systems to learn. Someone pointed out that people in distributed systems work are now using modal operators of the type Shoham is using in AOP.

2.4 Session 4: Integrative Approaches

2.4.1 Overview by Nils Nilsson

Nilsson discussed a sequence of architectures for integrated systems starting with one for a simple reactive system. He then elaborated this one to include first, learning, then state or a model of the world, then finally a model of the system's own actions so that the it could plan. His final architectural diagram is illustrated below:



In this scheme, all sensory signals are processed by various perception mechanisms and deposit information in a world model. The world model can contain both sentential and iconic information. An action-computation mechanism computes an appropriate next action based on the present state of the model. This action is then automatically executed regardless of what the planning and learning components might be doing. The action computation mechanism favored by Nilsson uses production rules whose condition parts are constantly being computed (based on what is in the model) and whose action parts specify durative (non-discrete) actions, which can be organized hierarchically [Nilsson 92]. Some of the actions may make changes to the model, so this scheme is a Turing-equivalent computational formalism. There is, of course, some leeway about how much computation is performed in the perception part and how much in the action part.

The planning sub-system uses information in the model (including a model of the agent's actions) to deduce what action ought (rationally) to be performed to achieve the agent's goals. Interaction between the planner and the action component is achieved by modifying the action component appropriately so it happens to compute the action specified by the planner. Of course, this modification can take place only if can be done *in time*, otherwise the system executes that action that the action component otherwise would have computed.

Learning operations can affect several parts of the system. They can change the way the planner operates, they can modify the model, and they can modify the action computation mechanism. Nilsson claimed that several seemingly different architectures can be viewed in this format.

2.4.2 Learning Agents

Presentation by Tom M. Mitchell

Abstract

One long-term goal of Artificial Intelligence is creating intelligent agents: sensor-effector systems that successfully achieve goals in complex environments. Different researchers take

different approaches to moving toward this goal. Some take a reductionist approach, attempting to solve one subproblem (e.g., vision, or natural language, or search), under the assumption that solutions to these subproblems will eventually be assembled into a more complete solution. Some take a theoretical approach, others experimental. Here I advocate pursuing the goal of intelligent autonomous agents by developing a sequence of architectures to support increasingly sophisticated agents. In our own research we are following this approach, guided by the following principles:

- first develop a simple, but complete, agent architecture, then elaborate it to handle more sophisticated tasks in more complex environments
- evaluate each architecture by its ability to *learn* to achieve its goals, (as opposed to a human's ability to program it to achieve these goals)
- learning mechanisms are primarily inductive (knowledge-driven learning occurs as an elaboration to a fundamentally inductive learning method)
- the agent may be initially given only certain knowledge: that which the architecture could, in principle, learn
- use the simplest possible set of architectural features, elaborating the architecture only as necessary

This research paradigm is similar to that of Brooks', Nilsson's, Hayes-Roth's, and the Soar effort, in that it attempts to develop complete agents rather than taking a reductionist approach. It differs from Brooks' and Hayes-Roth's in that our main focus is on architectures that support *learning* successful behavior, rather than manually programming such behaviors. This is because we believe learning capabilities provide a very strong constraint for pruning the space of possible architectures, and should therefore be considered right from the beginning. It differs from the work on Soar, in that we assume the primary learning method is inductive, whereas Soar assumes the primary learning method is chunking (an analytical, truth-preserving, explanation-based learning method). Our argument here is that explanation-based learning is unlikely to be the fundamental learning mechanism in an agent that is born with little initial knowledge (and therefore little that it can explain). Instead, we believe an inductive learning mechanism is more likely to be fundamental, with explanation-based learning incorporated as an elaboration to this basic inductive method.

One example of the type of research we are doing within this paradigm is our recent work on developing an architecture that enables the agent to combine such an inductive learning method with explanation-based learning. In this case, the learning task is to learn a control function that maps from the observed world state to the desired action for the agent (in fact, we formulate this as a reinforcement learning, or Q-learning task, as described in Barto's presentation at this workshop). An inductive method, neural network backpropagation, is used by the agent to learn a control strategy, based on training examples it collects by performing sequences of actions in its environment and observing their eventual outcomes. This basic inductive learning method is augmented by an explanation-based learning component that uses previously learned knowledge to significantly reduce the number of training experiences the agent must observe in order to learn an effective control strategy. This combined method is called explanation-based neural network learning (EBNN) [Mitchell 93]. The property desired of EBNN is that it be robust to errors in the agent's prior knowledge. In other words, we would like for it to leverage its prior knowledge when this knowledge is highly reliable, and to rely more strongly on the inductive learning component as the quality of its prior knowledge degrades. In experiments using a simulated robot domain, we observed exactly this property: when using very approximate initial knowledge the agent relied primarily on its inductive learning component, whereas it required fewer and fewer training examples as the quality of prior knowledge improved.

Discussion led by Yoav Shoham

Shoham asked if neural networks would also be useful in a software (as opposed to a robot) domain. Since sensing is cleaner and less problematic in the software domain, the generalizing abilities of neural nets might not be needed there. Victor Lesser wondered about mechanisms for accomodating sentential knowledge in Mitchell's learning agents. David Rumelhart suggested that declarative information might be stored in a neural-network-based associative memory and then retrieved when it was needed (to develop action models, for example).

2.4.3 Integrated Agents

Presentation by John E. Laird

Abstract

As a field, AI has pursued a strategy of divide and conquer as it has attempted to build systems that demonstrate intelligent behavior. This has led to the balkanzation of AI into subareas such as planning, knowledge representation, machine learning, machine vision, natural language, and so on. Research in each of these areas has been very successful. Unfortunately, the final (and usually most difficult part) of divide and conquer is the compose, or merge where the independent solutions must be integrated together. We find that little is known about integrating the results from AI's subfields to create integrated autonomous agents. Such agents must: interact with a complex and dynamic environment using perception and motor systems, react quickly to relevant changes in its environment, attempt a wide variety of goals, use a large body of knowledge, plan, learn from experience, use tools, use specialized forms of reasoning such as spatial reasoning, and communicate in natural language.

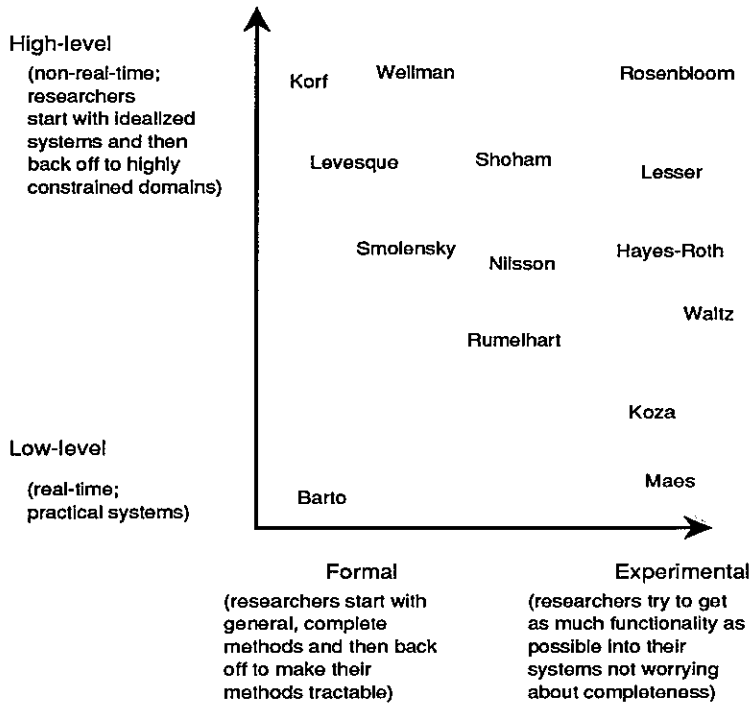
Our own research is aimed at trying to construct integrated agents and to determine the role of the underlying architecture in supporting the required capabilities. For example, what are the constraints on architectural learning mechanisms that can learn automatically about all tasks and subtasks—from skill acquisition, to concept acquisition, to learning from instruction? Our architecture of choice is Soar, which has already demonstrated significant integration, but not in situations requiring close integration with complex, external environments.

Soar is a promising architecture because of its multiple levels of control: the transduction level provides real-time encoding and decoding of input and output; the association level provides fast associative retrieval on knowledge, which is implemented as a parallel production system; the knowledge-based control level, which integrates global knowledge through repeated access of the associational level to select operations to perform both internally and externally; and the deliberation level, which allows unlimited, but controlled, planning and reflection, and is invoked automatically whenever impasses arise during knowledge-based control. Learning automatically converts knowledge that can be used flexibly by deliberation into situation-based associations that provide very fast responses.

In this talk we examine Soar on three different tasks, each of which stresses different aspects of integrated agent construction. The first task is mobile robot control. In this task, perception is incomplete, uncertain, asynchronous, errorful, and time dependent. Similarly, actions are conditional, time dependent, unpredictable, and errorful. The second task is simulated airplane control. In this task, the system must react quickly at many different levels as unexpected events occur in its environment. The third task is situated instruction of a simulated robot. This task, the system must understand natural language, as well as acquire new operators through instruction. For each of these tasks, we examine the contributions of the different architectural components of Soar (associational memory, impasse-driven subgoals, experience-based learning, etc.) to supporting the required capabilities and their integration.

3 Wrap-up Discussion

In a summary session near the end of the workshop, David Rumehart proposed yet another way to organize the different approaches represented. He also attempted to array them in two dimensions, namely 1) whether the approach was "high-level" or "low-level," and 2) whether the approach drew mainly from formal or from experimental methods. He used the following diagram:



There was, of course, some discussion about where participants ought to be placed on this chart. (Since Rumelhart presented the chart before Laird's and Mitchell's presentations, they are not on it.) Laird observed that the "Formal—Experimental" axis might be the same as the "Framework—Symbol—Implementation" dimension that he used to characterize the Soar work. In correspondence about Rumelhart's chart occurring after the workshop, Korf observed:

"I guess everyone will quibble with their place in the figure ..., so here's my quibble. If I take my name to represent a chess program, which is probably the canonical AI search program, it is characterized as high-level, non-real-time, and formal (general, complete) as opposed to experimental. The high-level characterization of the task is certainly accurate, but these programs are very much real-time systems, to the extent that they play in real tournaments with real time clocks. My suggestion is that the real-time issue is an orthogonal characterization, and should be stricken. As another example, most connectionist work is very low level, but far from real-time, since one simulates huge networks much more slowly than they actually work. On the formal-experimental scale, minimax search is a general, complete algorithm, given infinite computation, but there are no theorems worth knowing about chess programs. The domain is too complicated, and the behavior of depth-limited minimax too poorly understood. Rather, this is highly experimental work in that one builds real programs and runs them to see how well they play."

Vic Lesser concluded that the workshop demonstrated to him that there was a dominant "return-

to-weak-methods" movement within AI. Furthermore, researchers nowadays seemed much more willing to adopt an eclectic attitude toward system building rather than insisting that a single monolithic approach inform their efforts. John Laird thinks that there is a "return-of-tbe-agent" movement in AI (harking back, perhaps, to the work at SRI on Shakey the Robot in the late 1960s). The question now is how can researchers from the many requisite subfields come together to build useful agents.

David Waltz noted that the source of problems being attacked, whether they come from the real world or from artificial worlds, has a significant influence on approach. Nils Nilsson thought that researchers were motivated by more than simply the problems they were attacking; some are also driven by a desire to invent new generally useful techniques or broad principles. On that point, Paul Smolensky wondered whether anyone could name instances in which formal methods were invented first and then successfully applied rather than the needs of a specific application driving the invention of methods.

Vic Lesser concluded that, insofar as integrated agents are concerned, each component must be developed in close association with the other components rather than simply assembling standard ones developed in isolation. On a similar point, Dave Rumelhart thought that the workshop displayed a wide range of approaches that we all might be able to borrow from each other. He wondered whether there might be other important approaches not adequately represented at the workshop.

The National Science Foundation sponsor of the workshop, Dr. Su-shing Chen concluded that the workshop had been very successful. He thought that perhaps at the proper time it would be appropriate to have a follow-on workshop. Better yet, through the medium of electronic mail and the Internet, discussion of the various approaches to AI could be continuous among the present participants and other interested parties. He would particularly like to see discussions on future results, applications, research directions, and relationships among the approaches. He noted that some areas, for example machine learning and case-based reasoning, were not adequately represented at this workshop.

As a first step toward electronic discussion, this workshop report is being made available through anonymous ftp from the Santa Fe Institute. Directions for retrieving the LaTeX file or a Postscript version are given in Appendix B. Also an e-mail mailing list of all the participants of this workshop has been set up at the Santa Fe Institute. To send mail to each participant, address to: ai@santafe.edu.

Appendices

A Names and Affiliations of Participants

Richard E. Korf
Computer Science Department
University of California, Los Angeles
korf@cs.ucla.edu

Hector Levesque
Department of Computer Science
University of Toronto
hector@ai.toronto.edu

Barbara Hayes-Roth
Knowledge Systems Laboratory
Department of Computer Science
Stanford University
bhr@hpp.stanford.edu

Paul Rosenbloom
Information Sciences Institute and Computer Science Department
University of Southern California
rosenbloom@isi.edu

Paul Smolensky
Department of Computer Science
University of Colorado
smolensky@cs.colorado.edu

John R. Koza
Department of Computer Science
Stanford University
koza@cs.stanford.edu

Pattie Maes
Media Laboratory
Massachusetts Institute of Technology
pattie@media-lab.media.mit.edu

Andrew G. Barto
Department of Computer Science
University of Massachusetts
barto@cs.umass.edu

Victor Lesser
Department of Computer Science
University of Massachusetts
lesser@cs.umass.edu

Michael P. Wellman
Artificial Intelligence Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan
wellman@engin.umich.edu

Dave Waltz
Thinking Machines Corporation and
Brandeis University
waltz@think.com

Yoav Shoham
Robotics Laboratory
Department of Computer Science
Stanford University
shoham@cs.stanford.edu

Tom M. Mitchell
School of Computer Science
Carnegie Mellon University
tom.mitchell@cs.cmu.edu

John E. Laird
Artificial Intelligence Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan
laird@caen.engin.umich.edu

David E. Rumelhart
Department of Psychology
Stanford University
der@psych.stanford.edu

Nils J. Nilsson
Robotics Laboratory
Department of Computer Science
Stanford University
nilsson@cs.stanford.edu

Melanie Mitchell
Santa Fe Institute
mm@santafe.edu

Stephanie Forrest
Department of Computer Science
University of New Mexico
der@psych.stanford.edu

Su-shing Chen
IRIS
National Science Foundation
schen@nsf.gov

B Obtaining Copies of this Report via Anonymous FTP

This report is available over the Internet through anonymous ftp. One can get either the Postscript version of the file (approaches.ps) or the LaTeXversion (approaches.tex). If you get the LaTeXversion, you will also need the figures that go along with it and the bibliography file. The figures are in the files: architecture.eps, map1.eps, and map2.eps; the bibliography is in the file approaches.bbl.

The instructions for getting files using anonymous ftp are as follows:

ftp to santafe.edu (192.12.12.1)
log in as anonymous.
use your email address for a password.
type: "cd /pub/Users/mm/approaches"
type: "ls" to see what files are there.
get the desired files using the ftp 'mget' command.

References

[Barto 91] Barto, A., Bradtke, S., and Singh, S., "Real-time learning and Control Using Asynchronous Dynamic Programming," Technical Report 91-57, Computer Science Dept., University of Massachusetts, Amherst, MA, 1991.

[Feigenbaum 63] Feigenbaum, E., and Feldman, J., *Computers and Thought*, McGraw-Hill, New York, 1963.

[Halpern 90] Halpern, Joseph Y., "An Analysis of First-Order Logics of Probability," *Artificial Intelligence*, **46**, 311-350, 1990.

[Hayes-Roth 85] Hayes-Roth, B., "A Blackboard Architecture for Control," *Artificial Intelligence*, **26**:251-321, 1985.

[Holland 75] Holland, J., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975. (Second edition published in 1992.)

[Kitano 91] Kitano, H., *et al.*, "Massively Parallel Artificial Intelligence," *Proc. 12th International Joint Conference on Artificial Intelligence*, pp. 557–562, Morgan Kaufmann, San Mateo, CA, 1991.

[Korf 88] Korf, R. E., "Search in AI: A Survey of Recent Results," in *Exploring Artificial Intelligence*, H.E. Shrobe (Ed.), Morgan-Kaufmann, Los Altos, CA, pp. 197-237, 1988.

[Korf 92] Korf, R. E., "Search," in *Encyclopedia of Artificial Intelligence, Second Edition*, John Wiley, New York, pp. 1460-1467,1992.

[Koza 92] Koza, J. R., *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, 1992.

[McCarthy 58] McCarthy, J., "Programs with Commonsense," reprinted in Minksy, M., (ed.), *Semantic Information Processing*, pp. 403-418, MIT Press, Cambridge, MA, 1968.

[Maes 93a] Maes, P., and Kozierok, R., "Learning Interface Agents," *Proc. Nat'l Conf. on Artificial Intelligence*, American Assoc. for Artificial Intelligence, Menlo Park, CA, 1993.

[Maes 93b] Maes, P., *Proc. Second Conf. on Adaptive Behavior*, MIT Press, Cambridge, MA, February, 1993.

[Malone 88] Malone, T. W., Fikes, R. E., Grant, K. R., and Howard, M. T., "Enterprise: A Market-like Task Scheduler for Distributed Computing Environments," in B. Huberman (ed.), *The Ecology of Computation*, North-Holland, 1988.

[Mitchell 93] Mitchell, T., and Thrun, S., "Explanation-based Neural Network Learning for Robot Control," in Moody, J., Hanson, S., and Lippmann, R., (eds.), *Advances in Neural Information Processing Systems 5*, Morgan Kaufmann, San Mateo, CA, 1993.

[Newell 76] Newell, A., and Simon, H. A., "Computer Science as Empirical Inquiry: Symbols and Search," *Comm. of the ACM*, **19**(3): 113-126, 1976.

[Nilsson 92] Nilsson, N. J., "Toward Agent Programs with Circuit Semantics," Department of Computer Science Report No. STAN-CS-92-1412, Stanford University, Stanford, CA 94305, January 1992.

[Pearl 87] Pearl, J. and Korf, R. E., "Search Techniques," in *Annual Review of Computer Science*, **2**, Annual Reviews Inc., Palo Alto, CA, pp. 451-467,1987.

[Pearl 88] Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA, 1988.

[Rosenbloom 93] Rosenbloom, P. S., Laird, J. E., and Newell, A., *The Soar Papers: Research on Integrated Intelligence* (two volumes), MIT Press, Cambridge, MA, 1993.

[Savage 72] Savage, L. J., *The Foundations of Statistics*, Dover Publications, New York, 1972 (Second Edition).

[Selman 92] Selman, B., Levesque, H., and Mitchell, D., "A New Method for Solving Hard Satisfiability Problems," *Proc. Natl. Conference on Artificial Intelligence*, pp. 440-446, American Assoc. for Artificial Intelligence, Menlo Park, CA, 1992. .

[Shannon 50] Shannon, C.E., "Programming a Computer for Playing Chess," *Philosophical Magazine*, **41**, pp. 256-275, 1950.

[Shoham 92] Shoham, Y., and Tennenholtz, M., "Emergent Conventions in Multi-Agent Systems," in *Proceedings Symposium on Principles of Knowledge Representation and Reasoning*, 1992.

[Shoham 93a] Shoham, Y., "Agent Oriented Programming," *Artificial Intelligence*, **60**, 1:51-92, 1993.

[Shoham 93b] Shoham, Y., and Tennenholtz, M., "Computational Social Systems: Offline Design," *Artificial Intelligence*, to appear.

[Wellman 91] Wellman, Michael P, and Doyle, Jon, "Preferential Semantics for Goals," *Proc. Natl. Conference on Artificial Intelligence*, pp. 698-703, American Assoc. for Artificial Intelligence, Menlo Park, CA, 1991.

[Wellman 93a] Wellman, Michael P., "Challenges of Decision-Theoretic Planning," in *Working Notes of AAAI Spring Symposium on Foundations of Automatic Planning*, pp. 156-160, American Association of Artificial Intelligence, Menlo Park, CA, 1993.

[Wellman 93b] Wellman, Michael P., "A Market-Oriented Programming Environment and its Application to Distributed Multicommodity Flow Problems," *Journal of Artificial Intelligence Research*, **1**, 1993.

[Wilson 91] Wilson, S., "The Animat Path to AI," in Meyer, J. A., and Wilson, S. (eds.), *From Animals to Animats; Proceedings of the First International Conference on the Simulation of Adaptive Behavior*, The MIT Press/Bradford Books, Cambridge, MA, 1991