# Modeling Adaptive Autonomous Agents

Pattie Maes
MIT Media-Laboratory
20 Ames Street Rm 305
Cambridge, MA 02139
pattie@media.mit.edu

**Abstract**

One category of researchers in artificial life is concerned with modeling and building so-called adaptive autonomous agents. Autonomous agents are systems that inhabit a dynamic, unpredictable environment in which they try to satisfy a set of time-dependent goals or motivations. Agents are said to be adaptive if they improve their competence at dealing with these goals based on experience. Autonomous agents constitute a new approach to the study of artificial intelligence (AI) which is highly inspired by biology, in particular ethology, the study of animal behavior. Research in autonomous agents has brought about a new wave of excitement into the field of AI. This paper reflects on the state of the art of this new approach. It attempts to extract its main ideas, evaluates what contributions have been made so far and identifies its current limitations and open problems.

## 1 Introduction

Since 1985, a new wave has emerged in the study of artificial intelligence (AI). At the same time 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 "autonomous agent research", or "behavior-based AI" as opposed to main-stream "knowledge-based AI", or also "bottom-up AI" versus "top-down AI". Finally, the term "animat approach" (shorthand for "artificial animal"), which was coined by Wilson [69], is also frequently used.

Several people have given definitions and written overviews of research in Autonomous Agents, among others Brooks [11], Wilson [70] and Meyer [46]. There are several reasons for giving it yet another try. First of all many researchers are still skeptical about the approach. Some claim that it isn't very different from what they have been doing all along. Others are still not convinced that the approach is founded and scientific.

A second reason is that this account is different from the papers listed above. Brooks, being one of the main originators of this new approach, presents a picture which is restricted to robotic forms of intelligence [11]. This paper presents a more general perspective. It argues that the autonomous agent approach is appropriate for the class of problems that require a system to autonomously fulfill several goals in a dynamic, unpredictable environment. This includes applications such as virtual actors in interactive training and entertainment systems [3] [39], interface agents [40] [53], process scheduling [42], and so on. Wilson's account [70] focuses on a scientific methodology for research in autonomous agents, while Meyer [46] aims to give an overview of the research performed so far.

Finally, a third reason is that, since the approach has been around for a number of years now, it is time to perform a critical evaluation. This paper discusses the basic problems of research in adaptive autonomous agents. It also presents an overview and evaluation of the state of the art of the field. In particular it identifies some of the more general and more specific open problems that still remain to be solved. Overview papers are necessarily biased. This paper is biased toward the research in adaptive autonomous agents that has taken place at the AI Laboratory and Media Laboratory of the Massachusetts Institute of Technology.

The paper is structured as follows. Section 2 introduces the concept of an adaptive autonomous agent and defines the basic problems the field is trying to solve. Section 3 discusses the guiding principles of research in adaptive autonomous agents. Section 4 identifies the common characteristics of solutions that have been proposed. Section 5 discusses some example state of the art agents stemming from three different application domains: mobile robotics, interface agents and scheduling systems. Section 6 presents a critical overview of the state of the art. It discusses the main architectures that have been proposed for building agents. In particular, it addresses progress made in models of action selection and models of learning from experience. Section 7 presents some overall conclusions.

# 2 What is an Adaptive Autonomous Agent?

An *agent* is a system that tries to fulfill a set of goals in a complex, dynamic environment. An agent is situated in the environment: it can sense the environment through its sensors and act upon the environment using its actuators. An agent's goals can take many different forms: they can be "end goals", or particular states the agent tries to achieve; they can be a selective reinforcement or reward that the agent attempts to maximize; they can be internal needs or motivations that the agent has to keep within certain viability zones, and so on. An agent is called *autonomous* if it operates completely autonomously, i.e. if it decides itself how to relate its sensor data to motor commands in such a way that its goals are attended to successfully. An agent is said to be *adaptive* if it is able to improve over time, i.e. if the agent becomes better at achieving its goals with experience. Notice that there is a continuum of ways in which an agent can be adaptive, from being able to flexibly adapt to short-term, smaller changes in the environment, to dealing with more significant and long-term (lasting) changes in the environment, i.e. being able to change and improve behavior over time.

Depending on what type of environment it inhabits, an agent can take many different forms. Agents inhabiting the physical world are typically robots. An example of such an agent would be an autonomous vacuuming robot. Agents inhabiting the "cyberspace" environment consisting of computers and networks are often called "software agents" or "interface agents" or sometimes also "knobots". An example of such an agent would be a system that navigates computer networks to find data of a particular nature. Finally, agents can inhabit simulated physical environments. An example of such an agent could be a "synthetic actor" in a computer animated world. Combinations of these three types of agents may exist. For example, in the ALIVE interactive environment[39], the animated (virtual) agents employ real sensors (namely a camera), to decide how to react to a person's movements and gestures.

Even though artificial intelligence aims to study intelligence by synthesizing artificially intelligent systems, mainstream AI has so far concentrated on problems that are very different than that of modeling adaptive autonomous agents. Some key points which distinguish traditional AI from the study of autonomous agents are:

1. Traditional AI has focussed on systems that demonstrate isolated and often advanced competences (e.g. medical diagnosis, chess playing, etc). Traditional AI systems provide "depth" rather than "width" in their competence. In contrast, an autonomous agent has multiple integrated competences. Typically the competences are lower-level competences. For a robot these are competences such as locomotion, navigation, keeping the battery charged, collecting objects, etc. For other systems these might be other simple competences, like reacting in a market system by simple bidding and buying behaviors [42] or executing a simple software routine in the case of an interface agent [40].

2. Traditional AI has focussed on "closed" systems that have no direct interaction with the problem domain about which they encode knowledge and solve problems. Their connection with the environment is very controlled and indirect through a human operator. The operator recognizes a problem in the domain and describes it to the system in the symbolic language that the system understands. The system then returns a symbolic description of an answer or solution, which then has to be implemented by the operator in the actual domain. In contrast, an autonomous agent

is an "open" system. An agent is "situated" in its environment. It is directly connected to its problem domain through sensors and actuators. It can affect or change this domain through these actuators. The problem domain is typically very dynamic, which means that the system has a limited amount of time to act and that unpredictable events can happen. It typically also incorporates other acting agents (human and/or artificial).

3. Most traditional AI systems deal with one problem at a time. The problem the system has to solve is presented to the system by the human operator. Often, the system does not have time constraints for solving the problem and does not have to deal with interrupts (although the operator might have to deal with such problems). From the system's point of view the problem domain does not change while the system is computing. In contrast, an agent is autonomous: the system is completely self-contained. It has to monitor the environment and figure out by itself what the next problem or goal to be addressed is. It has to deal with problems in a timely fashion. Typically an agent has to deal with many conflicting goals simultaneously.

4. Traditional AI focuses on the question of what knowledge a system has. AI systems have declarative "knowledge structures" that model aspects of the domain of expertise. All of the internal structures, apart from an interpreter, are static. The system is only active when a problem is posed by the human operator, in which case the interpreter uses the static knowledge structures to determine the solution to the problem. In contrast, the emphasis in autonomous agent research is on what behavior a system demonstrates when put into its environment. The internal structures of an agent are dynamic "behavior producing" modules as opposed to static "knowledge structures". They do not have to be initiated by a goal formulated by a user. It is less important that the agent can answer questions about its problem domain (such as how it solves a particular problem). It is also less important that the user is able to inspect the internal structures and identify those that are responsible for particular aspects of the resulting behavior. For example, it is acceptable for goals or plans to be emergent observable properties that cannot be attributed to one particular internal structure (but that instead are the result of some complex interaction among a set of structures and the environment).

5. Finally, traditional AI is not usually concerned with the developmental aspect or the question of how the knowledge structures got there in the first place and how they should change over time. They do not have to be adaptive to changing situations (components breaking down, etc). Most of the work done in traditional machine learning assumes that a lot of background knowledge is available. This background knowledge is used by the system to do knowledge reformulation or knowledge compilation (e.g.

cacheing, explanation based learning, concept learning, etc). In contrast, in autonomous agent research there is a strong emphasis on "adaptation" and on a "developmental approach". This often means that the system improves its own internal structures (and thus its behavior) over time, based on its experience in the environment. The agent actively explores and updates its structures using an incremental, inductive learning method. In other cases, this means that the designer takes an incremental approach to building the agent: the user gradually evolves a more sophisticated system by adding structure to an already existing "working" system.

The main problem to be solved in autonomous agent research is to come up with an architecture for an autonomous agent that will result in the agent demonstrating adaptive, robust and effective behavior. *Adaptive* means that the agent improves its goal-achieving competence over time. *Robust* means that it never completely breaks down (it demonstrates graceful degradation when components within the agent fail or when unexpected situations happen). *Effective* means that the agent is successful at eventually achieving its goals. Specifically, two related subproblems have to be solved:

1. *The problem of action selection:*
   How can an agent decide what to do next so as to further the progress towards its multiple time-varying goals? How can it deal with contingencies or opportunities that may arise? How can it arbitrate among conflicting goals? How can it deal with noisy sensors and actuators? How can it react in a timely fashion? etc.

2. *The problem of learning from experience:*
   How can an agent improve its performance over time based on its experience? How can it decide when to "exploit" its current best action, versus "exploring" other actions so as to possibly discover better ways of achieving its goals [21]? How can it incorporate the feedback from the world into its internal behavior-producing structures? How can it correct "wrong" or ineffective behavior-producing structures? etc.

Section 6 discusses both of these problems in more detail. In summary, the main goal of research in autonomous agents is to better understand the principles and organizations that underlie adaptive, robust, effective behavior. A secondary goal is to also develop tools, techniques and algorithms for constructing autonomous agents that embody these principles and organizations. We call the totality of a set of principles, an organization and the set of tools, algorithms and techniques that support them, an "architecture" for modeling autonomous agents.

One of the few things that has become clear in the last couple of years is that there does not exist one such architecture that can be considered optimal in all respects (or better than all the other ones proposed). Rather, the goal of our research has become to develop an understanding of which architectures are the most simple solution for a given class of agent problems. More specifically such a problem class is defined in terms of particular characteristics of the agent's resources (e.g. memory, sensors, compute power), and particular characteristics of the tasks and environment [64] [32].

# 3 Guiding Principles

The study of Adaptive Autonomous Agents is grounded in two important insights. These serve as "guiding principles" for the research performed:

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

The first realization is that viewing the problem of building an intelligent system in its context can make things a lot easier. This observation is true at several levels:

1. The intelligent functions that are being modeled, such as perception, planning, learning, etc, are part of a complete intelligent system, namely the agent. Building systems in an integrated way rather than developing modules implementing these functions independently, often makes the task a lot easier. For example, a system that can learn can rely less on planning because it can cache computed plans for future reuse. A system that has sensors and actuators can perform tests in the environment and as such has less of a need for modeling its environment and for inference and reasoning. A system that has sensors has an easier job disambiguating natural language utterances, because most likely they are related to the objects the system currently perceives, and so on.
2. A complete intelligent system is always part of some environment; it is situated in some space. This implies that there is less of a need for modeling, because the "world is its own best model" [11]. The environment can also be used as an external memory, for example, for reminding the system which tasks still have to be performed and which ones it already did perform [59]. The environment usually has particular characteristics that can be exploited by the system. For example, offices consist of vertical walls and horizontal floors, doors typically are of a particular size, etc. These "habitat constraints" can be exploited by the system, making its task much easier [22].
3. An intelligent system is not only situated in space, but also in time. This implies that the system can develop itself so as to become better at its task,

if time and the particular task permit (through learning from experience). Time also allows for the construction of an iterative, incremental solution to a problem[1]. For example, a natural language system situated in time does not need to be able to disambiguate every utterance. It can engage in a discourse, e.g. asking questions or making particular remarks that will help it to gradually disambiguate whatever the other speaker wants to convey.

4. Finally, every intelligent system is typically also part of a society. Other agents in the same environment are dealing with similar or related problems. Therefore there is no need for the agent to figure everything out by itself. For example, a mobile robot could use the strategy of closely following a person passing by, in order to achieve the competence of navigating in an office environment without bumping into things. Maes and Kozierok [40] report on some experiments in which interface agents learned to perform certain tasks by observing and imitating users.

As a consequence of the above ideas, autonomous agent research has concentrated on modeling systems within their context. Except for expert systems research, traditional AI has concentrated on more abstract and hypothetical problems, while Behavior-Based AI or agent research has built "real" systems that solve an actual (small) problem in a concrete environment.

A second major insight upon which the study of Autonomous Agents is founded is that interaction dynamics among simple components can lead to emergent complexity (see also [51]). Agent research is founded on the belief that shifting into the "interaction" domain as opposed to the "component" domain will make it easier to solve the problem of building intelligent systems. This idea also applies at several different levels [12]:

1. Interaction dynamics between an agent and its environment can lead to emergent structure or emergent functionality. This idea is inspired by the field of ethology. Ethologists have stressed that an animal's behavior can only be understood (and only makes sense) in the context of the particular environment it inhabits. Braitenberg, a cybernetician, also convincingly illustrated a similar idea in his book "Vehicles" [9]. Finally, in AI, Simon [55] referred to the same idea when he discussed the example of an ant on the beach. He notes that the complexity of the ant's behavior is more a reflection of the complexity of the environment than of its own internal complexity. He muses that one could think that this is true for human behavior. Many years later, Agre [1] showed how behavior as complex as goal-directed action sequences can be modeled as an emergent property

[1] Situatedness in time cuts both ways: it also means that the agent has to react in a timely fashion and be able to deal with interrupts.

of the interaction dynamics between a complex environment and a reflex-guided agent. What this all means is that the internal structures controlling an agent need not be complex to produce complex resulting behavior. It is often sufficient to study the particular properties of the environment and find an interaction loop, a set of feedback or reflex mechanisms that will produce the desired behavior. One consequence is that we need a better understanding of the particular characteristics of an environment. If we want to be able to understand or prove aspects about the resulting performance of autonomous agents, we have to model the agent as well as its environment [5] [23]. Another consequence is that we need better models of the interaction dynamics between an agent (or components of the agent) and its environment.

2. Simple interaction dynamics between the different components within an agent can lead to emergent structure or emergent functionality. For example, Mataric's wall-following robot does not have a single component to which the expertise of wall-following can be attributed [43]. One module is responsible for steering the robot towards the wall when the distance to the wall is above some threshold while another module is responsible for steering the robot away from the wall when the distance is below some threshold. Neither one of these modules is primarily "responsible" for the wall following behavior. It is their interaction dynamics that makes the robot follow walls reliably. In Maes' networks [33], none of the component modules is responsible for action selection. The action selection behavior is an emergent property of some activation/inhibition dynamics among the primitive components of the system.

3. Interaction dynamics between the component agents of a social system can lead to emergent structure or functionality. Deneubourg [16] [17] describes how social insects following simple local rules can produce emergent complexity such as a path to a food source, food foraging trees, etc. Malone's collection of autonomous bidding systems addresses the complicated task of process-processor allocation [42]. Finally, anthropologists have studied how different concepts and complex methods for solving problems are gradually shaped through social interaction among different people [59] [54].

What is important is that such emergent complexity is often more robust, flexible and fault-tolerant than programmed, top-down organized complexity. This is the case because none of the components is really in charge of producing this complexity. None of the components is more critical than another one. When one of them breaks down, the system demonstrates a graceful degradation of performance. Since all of the components interact in parallel, the system is also able to adapt more quickly to environmental changes. Often the system explores multiple solutions in parallel, so that as soon as certain variables change, the system is able to switch to an alternative way of doing things. For example, in Maes' system [33] several sequences of actions are evaluated in parallel, the best one determining the behavior of the agent. Also in Malone's system [42] several mappings of processes to machines can be viewed as being explored in parallel.

# 4 Characteristics of Agent Architectures

Many of the architectures for autonomous agents that have been proposed have characteristics in common. This section lists these shared characteristics and contrasts them with the characteristics of traditional AI architectures. These differences are illustrated by means of some concrete examples in the next section.

## Task-Oriented Modules

In traditional AI, an intelligent system is typically decomposed along "functional modules" such as perception, execution, natural language communication (the peripheral components), a learner, planner and inference engine (the central systems components). These modules are typically developed independently. They rely on the "central representation" as their means of interface. The central representation includes things such as beliefs, which are updated by the perception component and processed and augmented by the inference engine and natural language component, desires (or goals) and intentions, which are produced by the planner.

In contrast, an agent is viewed as a set of competence modules (often also called behaviors) [10]. These modules are responsible for a particular small task-oriented competence. Each of the modules is directly connected to its relevant sensors and actuators. Modules interface to one another via extremely simple messages rather than a common representation of beliefs, and so on. The communication between modules is almost never of a "broadcast" nature, but happens rather on a one-to-one basis. Typically the messages consist of activation energy, or simple suppression and inhibition signals, or simple tokens in a restricted language. In addition to communication via simple messages, modules also communicate "via the environment". One module may change some aspect of the environment which will trigger another module, etc.

## Task-Specific Solutions

In traditional AI, the different functional components of the system are modeled as general and domain-independent as possible. The hope is that the same functional components can be used for different problem domains (a general domain-independent planner, learner, etc). The only component that needs to be adapted is the central representation, which contains domain-specific information such as a model of the particular environment at hand and possibly also more heuristic knowledge.

In contrast, an agent does not have "general" or task-independent functional modules. There is no general perception module, no general planner, etc. Each of the competence modules is responsible for doing all the representation, computation, "reasoning", execution, that is necessary for the particular competence it is responsible for. For example, an obstacle avoidance module might need one bit of information to represent whether an obstacle is perceived or not within a critical range. It might do some very simple computation to decide how an obstacle should be avoided. Competence modules are self-contained, black boxes. They might employ completely different techniques (even different hardware) to achieve their competence. Part of the reason for this more pragmatic approach is a pessimistic vision about whether it is possible at all to come up with a general solution to the vision problem, a general solution to the planning problem, etc, a view also expressed by Minsky [47].

### Role of Representations is De-emphasized

In traditional AI, the key issue emphasized is that the agent has a complete, correct internal model; an accurate copy of the environment (with all its objects and relationships) inside the system, that the system can rely on to predict how its problems can be solved.

In contrast, in agent research there is little emphasis on modeling the environment. First of all, there is no central representation shared by the several modules. The system also does not attempt to integrate the information from different sensors into one coherent, objective interpretation of the current situation. Instead, every task-oriented module locally represents whatever it needs to represent to achieve its competence. The localized representations of different modules are not related and might be inconsistent with one another or redundant. Within one competence module, the usage of representations may be minimized in favor of employing the environment as a source of information (and a determiner of action). The representations within one module are often of a less propositional, objective and declarative nature than those employed in traditional AI. For example they might index objects according to the features and properties that make them significant to the task at hand [1] rather than the identities of the objects. They can be of a numeric, procedural [44] or analog nature. Often a lot of task-specific "problem solving" is performed in the perception part of a particular competence [57] [14] [2].

### Decentralized Control Structure

Traditional AI adopts a sequential organization of the different modules within the system. The modules take turns being "active" or processing and changing the internal representations. Perception and inference first update the internal model (beliefs and goals). After that, planning or problem solving produce a description of the solution to the problem (a plan or the answer to a question). Finally, either the execution module or a human operator implements the solution in the domain (the latter one having more knowledge and understanding of the situation than the former one).

In contrast, agent architectures are highly distributed and decentralized. All of the competence modules of an agent operate in parallel. None of the modules is "in control" of other modules. However, some simple arbitration method is included in order to select or fuse multiple conflicting actuator commands (commands of different modules might be mutually exclusive). This arbitration network might be a winner-take-all network, as in [33] or a hardcoded priority scheme as in [10]. Because of its distributed operation, an agent is typically able to react quickly to changes in the environment or changes in the goals of the system.

### Goal-directed Activity is an Emergent Property

Traditional AI models activity as the result of a "deliberative thinking" process. The central system evaluates the current situation as represented in the internal model and uses a search process to systematically explore the different ways in which this situation can be changed so as to achieve a goal situation.

In contrast, in agents, activity is not modeled as the result of a deliberative process. Instead, complex and goal-directed activity is modeled as an emergent property of the interaction among competence modules internally, and among competence modules and the environment. There is no internal structure corresponding to "the plan" of the system. Many agents do not have any explicit goals, but are nevertheless still driven towards a specific set of fixed, compiled-in goals. In other architectures, the agent has an explicit representation of its (possibly time-varying) goals, which is used to modify the priorities among the different modules over time.

### Role for Learning and Development

In traditional AI learning typically consists of compilation or reformulation of what the system already knows. For example, the system might cache a plan for later reuse. Very seldom does the system perform inductive learning of new information or corrective learning of existing knowledge based on environmental experimentation and feedback. This implies that the programmer is completely responsible for creating an initial complete and correct model for the system to use.

In contrast, learning and development are considered crucial aspects of an adaptive autonomous agent [69]. Building an adaptive system that will develop from a not so successful system into one that achieves the tasks, is often considered a better approach than building a successful system that does not change when the environment or task changes (e.g. a robot breaking one of its legs). In some systems, the evolution towards increasingly more sophisticated and more adaptive behavior is simulated by the programmer, e.g. by incrementally adding more structure to existing successful systems [11]. Other systems employ learning by the individual [38] [35] [69] [18] [25] [62]. In almost all cases, the system concentrates on learning new information (or behavior) from its environment, rather than on reformulating information it already has. The learning algorithms are implemented in a distributed way: typically a similar learning algorithm runs in different competence modules. Related to the idea of learning is that of redundancy: often the system has multiple modules for a particular competence. Experience sorts out which of these modules implements the competence in a more reliable way and should thus be preferred [38] [50] [18].

Systems built using all of the above principles (task-oriented modules, task-specific solutions, de-emphasized representations, decentralized control, etc) tend to demonstrate more adaptive and robust behavior. They act (and react) quickly, because (1) they have fewer layers of information processing, (2) they are more distributed and often non-synchronized, and (3) they require less expensive computation (they are not prone to problems of combinatorial explosions, because they do not rely on search processes as much). They are able to adapt to unforeseen situations (opportunities as well as contingencies), because they rely much more on the environment as a source of information and a determiner of action than on their possibly faulty or outdated model. They are robust because (1) none of the modules is more critical than the others, (2) they do not attempt to fully understand the current situation (which is often time consuming and problematic), (3) they incorporate redundant methods and (4) they adapt over time.

# 5 Some Example Autonomous Agents

## A Mobile Robot

Consider a mobile surveillance robot that has to monitor some offices. Its task requires that it navigate from room to room. The traditional AI version of this robot could work in a similar way to Shakey [48]. The perception module processes the different sensor data and integrates them into a representation of the environment. It attempts to update this model as often as possible. The model includes information such as the location of the robot in the environment, the location and type (often even identity) of other objects in this environment such as chairs, tables, etc. The model is used by the planning module to decide how to fulfill the goal of finding the door in the current room, while avoiding obstacles. The planner goes through a systematic search to produce a list of actions that will according to the model fulfill both goals. The execution module executes this plan while possibly checking at certain points whether things are going as predicted. If not, control is returned to the planner.

An adaptive autonomous agent for the same task could be constructed in the following way (as inspired by[10]). In an incremental way, several modules would be implemented corresponding to the different competences necessary for the task: a module for recognizing and going through doors, a module for wall following, a module for obstacle avoidance (or even a couple of redundant ones, using different sensors, since this is a very critical competence), and so on. All of these modules operate in parallel. A simple arbitration scheme, for example suppression and inhibition wires among these modules, suffices to implement the desired priority scheme: the obstacle avoidance modules always have priority over going through doors, which has priority over wall following. This robot does not plan a course of action. However, from an observer's point of view it will appear to operate in a systematic, rational way. Brooks [10] [11] has argued convincingly, in writing and in demonstrations, which of the two above robots will be more successful at dealing with the task in a robust and reliable way[2].

## An Interface Agent

Consider the problem of building an intelligent autonomous system that helps the user with certain computer-based tasks. Its goal is to offer assistance to the user and automate as many of the actions of the user as possible. Traditional AI has approached this problem in the following way [60]. The system is given an elaborate amount of knowledge about the problem domain by some knowledge engineer: a model of the user and possibly the user's organization, a model of the tasks the user engages in, including a hierarchical specification of the subtasks, knowledge about the vocabulary of these tasks, and so on. At run-time, the agent uses this knowledge to recognize the intentions and plans of the user. For example, if a UNIX user enters a command like "emacs paper.tex", the system infers that the user is planning to produce a written document. It then plans its own course of action (the goal being to assist the user), which for example might consist of the action sequence: the text formatting command "latex paper.tex", followed by the preview command "xdvi paper.dvi" and the printing command "lpr paper.dvi". The problems with this approach are exactly the same ones as those of traditional AI robots: it is hard to provide such a complete and consistent model and the model is quickly outdated (as the user's ways of performing tasks change). Because of the computational complexity of the approach, the system would react very slowly. All sorts of unpredicted

[2] Ideally, the robot would also monitor the results of its actions and learn from experience so as to improve its competence or deal with significant changes in the robot or its environment, i.e. so as to demonstrate robust and effective autonomous behavior over longer periods of time.

events might take place that the system cannot deal with, for example, the user might change his/her mind about what to do in the middle of things, or might perform tasks in unorthodox or non-rational ways, etc.

Instead, an adaptive autonomous "interface agent" can be built as follows [40]. Several competence modules are constructed that are experts (or try to become experts) about a small aspect of the task. For example, one module might be responsible for invoking a particular command (like "lpr") at a particular moment. The agent is situated in an environment containing an ideal source for learning: the user's behavior. Each of the modules gathers information by observing the user and keeping statistics about a particular aspect of the user's behavior. For example, the above mentioned module will keep track of the situations in which the user executed the "lpr" command. Whenever a new situation comes up that is very similar to one of one or more memorized situations, it actually offers to the user to execute the "lpr" command. If we have several experts for the different commands listed above, each of these will know when to become active and offer their assistance to the user. From an observer's point of view, it will seem as if the system "understands" the intentions of the user, as if it knows what the task of producing a document involves. Nevertheless, the action sequences are just an emergent property of a distributed system. The system will smoothly adapt to the changing habits of the user, will react in a fast way, will be less likely to completely break down, and so on.

## A Scheduling System

Finally, consider the problem of building a scheduling system that has as goal to allocate processes to processors in real-time. Again the domain is a very dynamic one: new processing jobs are formulated in different machines all the time. The decision to be made is whether to run these processes locally or on a different machine, the global goal being to minimize the average amount of time it takes to run a process. The loads of the different available machines vary continuously. Certain machines might suddenly become unavailable for scheduling processes, requiring a rescheduling of the jobs that were running on those machines at the time, and so on. A traditional AI system for this task would contain a lot of knowledge about scheduling and about the particular configuration of machines and typical processing jobs at hand. The system would update its representation of the current situation as often as possible. This requires gathering all the data from the different machines in the network on whether they are still available, what their workload is, which processes they are running, which new processes were formulated on them, etc. Once all this information has been gathered, the system would perform a systematic search (possibly involving some heuristics) for the most optimal allocation of processes to processors. Once that schedule has been produced, the processing jobs can actually be sent to the different machines that they have been assigned to. This centralized way of solving the problem is present in the majority of the earlier work in this area [29].

Among others, Malone has proposed a different solution to this problem [42], that one could call more "agent-based". In his Enterprise system, each of the machines in the network is autonomous and in charge of its own work load. The system is based on the metaphor of a market. A machine on which a new processing task originates sends out a "requests for bids" for the task to be done. Other machines may respond with bids giving estimated completion times that reflect their speed and currently loaded files. For example, if the task to be performed is a graphics rendering job and some machine has that software loaded, it will make a better bid for the new job (because it does not have to waste time and space loading the necessary software). The machine that sent out the request for bids will collect the bids it receives over some small period of time and allocate the job to the machine that made the best bid (either remote or local). This distributed scheduling method was found to have several advantages. The system is very robust because none of the machines is more critical than another one (there is no central scheduler). A user can make a machine unavailable for processing external jobs at run-time. The whole system will adapt smoothly to this unexpected situation. The solution is simple and yet very flexible in terms of the different factors it can take into account.

# 6 Overview of State of the Art

Section 5 presented a general overview of the agent approach to building intelligent systems that demonstrate adaptive, robust behavior. This section provides a more detailed account of the specific architectures that have been proposed. In addition, it lists what the limitations and open problems are of the particular architectures proposed. Section 2 argued that there are two subproblems involved in modeling adaptive autonomous agents: the problem of action selection and the problem of learning from experience. This section is structured around these two subproblems. Most of the architectures for agents that have been proposed so far have concentrated on one or the other subproblem: either the agent combines simplistic action selection with sophisticated learning or it demonstrates sophisticated action selection without doing any learning. Few proposals have addressed both problems at once in the same architecture. In the remainder of this section a more detailed description of both of these subproblems is given, followed by a discussion of what progress has been made towards them, and a discussion of what questions remain unresolved.

## 6.1 Action Selection

### 6.1.1 The Problem

The problem of action selection can be stated as follows. Given an agent that has multiple time-varying goals, a repertoire of actions that can be performed (some of which are executable) and specific sensor data, what actions should this agent take next so as to optimize the achievement of its goals[3]. Notice that when we also consider learning from experience, this problem becomes a slightly different one because one of the goals of the agent is to learn how to better achieve its goals.

It is theoretically possibly to compute the optimal action selection policy for an agent that has a fixed set of goals and that lives in a deterministic or probabilistic environment [67]. What makes it impossible to do this for most real agents is that such an agent has to deal with (i) resource limitations (time, computation, memory), (ii) possibly incomplete and incorrect information (sensor data), (iii) a dynamic, non-deterministic, non-probabilistic environment, (iv) time-varying goals, (v) unknown and possibly changing probability distributions, and so on.

The goals the agent tries to satisfy can take many different forms: end-goals also called "goals of attainment" (end states to be achieved), negative goals (states to be avoided), needs, drives, desires, tasks, motivations, constraints for a plan, viability zones for certain state variables, etc. An agent typically has multiple conflicting goals. Being a "complete" system it always has a combination of "self-preservation goals" (e.g. not bump into obstacles, keep battery charged) as well as more task-oriented goals (watch over a set of offices). The goals of an agent can be implicit or explicit. In the former case, the agent does not have any explicit internal representation of the goals it is trying to achieve. The agent is built in such a way that, when situated in its environment, its behavior tends to achieve certain goals. Implicit goals are necessarily fixed. They cannot be changed unless the agent is reprogrammed. More complicated agents have explicit goals that vary over time and often have levels of intensity as opposed to a boolean on-off nature. For example, an artificial animal might have a particular hunger level, thirst level, etc.

Given that it is theoretically impossible to prove what the optimal action selection policy for an agent is, how does the field evaluate a particular proposed solution? Researchers in adaptive autonomous agents are not interested in provable optimality of action selection, i.e. in whether the agent takes the optimal path towards the goals, as they are in whether the action selection is robust, adaptive and whether the agent achieves its goals within the requirements and constraints imposed by the particular environment and task at hand. Among other issues this means that the action selection mechanism should:

- favor actions contributing to the goals, in particular, it should favor those actions that result in the most progress towards the goals,
- be able to flexibly deal with opportunities and contingencies,

[3] In this definition, one can substitute the word "competence module" or "behavior" for "action": given a set of competence modules that all try to control the actuators at a particular moment in time, which ones of those should be given priority, or how should their outputs be combined into one command for the actuators?

- be real-time (fast enough for the particular environment at hand and its pace of changes),
- minimize unnecessary switching back and forth between actions contributing to distinct goals,
- improve on the basis of experience (more on this in the next section),
- demonstrate graceful degradation when components break down or unexpected changes happen,
- never get completely stuck in a loop or deadlock situation, or, make the agent mindlessly pursue an unachievable goal,
- and most importantly, be "good enough" for the environment and task at hand: as long as the agent manages to achieve its goals within the constraints (time, quality, etc) required by the problem situation, the solution is considered an acceptable one. For example, as long as the robot manages to find the recharging station before its battery dies, as well as make sufficient progress towards its more task-specific goal of surveying the offices, it is considered an acceptable solution, even if it does not always follow optimal paths. Brooks [11] refers to this latter criterion as "adequacy".

McFarland [45] takes this last point even further. He argues that one should use an ecological approach to evaluate agent behavior: if an agent fills a market niche, then that agent is considered successful. That is, in McFarland's view, for agent behavior to be adaptive means that it must optimize its behavior with respect to the selective pressures of the market place. Even though this is ultimately true, it is not particularly useful as a means for comparing different proposals for agent architectures.

Tyrrell [67] compares several action selection proposals, but he does so with respect to one particular benchmark environment and task. Maes [36] and Wilson [70] have argued that it is not possible to decide that one action selection model is better than another one unless one also mentions what the particular characteristics are of the environment, the task and the agent. For example, in an environment where the cost of (incorrect) actions is high, an agent should do more anticipation; while if the cost of (incorrect) actions is neglectable, it does not matter if the agent often performs incorrect actions; in an environment where a lot of things change quickly, an agent needs to act very quickly; an agent with noisy sensors should have some inertia in its action selection so that one wrong sensor reading does not make the agent switch to doing something completely new and different; an agent with many sensors can rely on the environment to guide its selection of actions, while an agent with fewer sensors will need to rely more on its internal state or memory to decide what to do next. Todd and Wilson [64] and Littman [32] have started to build a taxonomy of environments and taxonomy of agents that will provide a more profound basis for comparing different proposals.

### 6.1.2 Progress Made

The different models for action selection in an autonomous agent that have been proposed differ in the way they deal with the following three problems:

1. What is the nature of the goals?
2. What is the nature of the sensor data?
3. What is the arbitration mechanism and command fusion mechanism?

The architectures proposed can be subdivided in the following three classes:

*Hand-built, Flat Networks.*

A number of architectures have been proposed that require the designer of the agent to solve the action selection problem from scratch for every agent that is built. Examples of such architectures are the Subsumption Architecture [10] and the architectures reported in [15] (a minimalist version of the Subsumption Architecture) [1] [14] [4] and others. All of these architectures require the designer of an agent to carefully analyze the environment and task at hand and then design a set of reflex modules and a way of combining the outputs of the modules (by means of suppression and inhibition wires or simple arbitration circuitry).

This class of architectures deals with the above three problems in the following way. Typically goals are implicit: they only exist (or may not even exist) in the designer's mind. An agent can have multiple goals and they can be of very different nature. The nature of the sensor data is also unlimited. The arbitration mechanism determining which modules will steer the actuators is implemented by a logical circuit or a set of suppression and inhibition wires. This circuit ensures that at most one module controls an actuator at all times. None of these architectures support command fusion. In other words, none of these action selection models make it possible for two or more modules to simultaneously determine what the command is that is sent to the actuators. For example, it is not possible to average the outputs of two modules.

One disadvantage of this class of solutions is that they don't offer the user much guidance as to how to solve the problem of action selection for a new agent. The architecture at most provides a philosophy, a set of previous successful examples and a programming language for building new agents. Another disadvantage of this class of solutions is that its solution does not scale up. For more complex agents the problem of action selection and arbitration among modules is too hard to be solved by hand. The arbitration network often becomes a complicated "spaghetti" that is hard to debug or to get to do the right thing. A final disadvantage is that most of these architectures do not allow for time-varying goals (because typically goals are not explicitly represented in the agent).

*Compiled, Flat Networks.*

A second class of architectures attempts to facilitate the construction of agents by automating the process of designing the arbitration circuitry among competence modules. Examples of such architectures are the Rex/Gaps system [24], Behavior Networks [33] and Teleo-Reactive Trees [49]. These architectures require the designer to specify in a particular formalism what the goals of the agent are, how goals can be reduced to other goals or to actions and what the different modules/actions are and their conditions and expected effects. A compiler analyzes this specification and generates a circuit that will implement the desired goal-seeking behavior.

In Kaelbling's and Rosenschein's work [24], the types of goals and sensors that can be dealt with are restricted to booleans. On the one hand this restricts the type of agent that can practically be built[4], but on the other hand these restrictions make it possible to prove that the circuitry synthesized will make the agent select the right actions so as to fulfill its goals. In most of the work, except for [33], these types of architectures produce agents with implicit, fixed (not time-varying) goals. However, in contrast with the previous class of architectures, the goals are explicit in the designer's formal specification of the agent. This implies that the agent's circuitry has to be resynthesized if the agent should fulfill a different set of goals.

Maes [33] [37] proposes an architecture with explicit, time-varying goals. The arbitration network that is compiled has an explicit representation of the goals of the agent and these goals can have intensities that vary over time (e.g. hunger level for an artificial animal, or motivation to recharge the battery of a robot). This particular system performs a limited form of arbitration, prediction and "planning" at run-time. More specifically, these processes are modeled in terms of a time-varying spreading activation process which makes activation energy accumulate in modules that are most relevant given the particular goals (and intensities) and sensor data at hand. Unfortunately in this system the sensor data are restricted to booleans.

One of the disadvantages of this class of action selection architectures is that the class of agents that can be built is restricted. This is the case because these architectures offer a particular model of action relevance, while the previous category does not impose any model at all. A second problem is that it is sometimes hard to come up with a declarative specification of the goals and desired behavior of an agent. Finally, the agent's action selection will only be as good as the specification it relies on. If the designer's specification of the effects of actions is erroneous, then the agent's behavior will not be as desired.

[4] At the least, it makes it more complicated to build certain kinds of agents.

*Hand-built, Hierarchical Networks.*

A final category of action selection models proposes a more hierarchical organization of the different actions or competence modules. Examples of such architectures are Agar [65], Hamsterdam [7], Rosenblatt and Payton's work [52], which is a more sophisticated version of the Subsumption Architecture and Tyrrell's work [67]. Most of these architectures are closely inspired by models of animal behavior stemming from ethologists such as Lorenz and Tinbergen. Typically these architectures organize actions in a hierarchy that ranges from high-level "modes" or activities via mid-level composite actions to detailed, primitive actions. Only the primitive actions are actually executable. Tyrrell [67] and Blumberg [7] both have demonstrated that when scaling the problem to more complex agents that have many different goals and actions, it is desirable to have more structure (than that present in flat networks) that may help decide which actions are relevant. Typically these systems use some sort of action selection at higher abstraction levels to prime or bias the selection of more primitive actions.

This last category of systems supports more complex (animal-like) motivations or goals. The model of the sensors and how they affect the action selection is also more sophisticated. For example, some architectures make it possible for a stimulus (sensor data) to have a certain "quality" or "intensity" that will affect the action selection (e.g. not just "is food present", but "is food present and what is the quality of the food stimulus perceived"?). As is the case with the first class of architectures discussed, these architectures require the designer to build the arbitration network by hand. Often this is a very difficult and tricky task, in particular because these ethology-based models tend to have a lot of parameters that need be tuned to obtain the desired behavior.

### 6.1.3 Open Problems

Even though a lot of progress has been made towards the study of action selection models, many problems remain unresolved:

- Very little research has been performed on the nature of goals and goal interactions. We need to study what kinds of goals our architectures need to support, where those goals might come from, how they change over time, etc. Toates and Jensen [63] present a nice overview of the different models of motivations that ethology and psychology have come up with.

- In most of the architectures proposed, scaling to larger problems is a disaster. This is especially so in the case of hand-built networks, because no support is given to the designer of an agent for building the complicated arbitration network that will govern its behavior. The most obvious solution to be investigated is to either evolve [13] or learn and adapt the network [35] based on experience. However, few experiments along these lines have been performed so far.

- Related to this, not enough effort has been put into making pieces of agent networks "reusable" within other agents. Given that the first and third category of architectures reduce the action selection problem to a (non-trivial) engineering problem, it would be useful if partial solutions that have proven to work in one agent could be abstracted and reused in another agent. For example, the modules producing wall-following behavior in one robot could be abstracted so that they can be reused in another robot with comparable sensors and a comparable environment.

- As noted by many other authors, the dynamics of interactions between the agent and its environment and among the different modules of one agent are not well understood. Kaelbling and Rosenschein offer a logical model [24], while Beer [5], Kiss [28] and Steels [58] have started approaching this problem from a dynamical systems perspective. Nevertheless, the field is far from being able to prove in general what the emergent behavior is of a distributed network of competence modules.

- Most of the proposed architectures do not deal with the problem of command fusion. Typically only one module at a time determines what command is sent to an actuator. There is no way for the outputs of multiple modules to be combined. Some proposals for solutions to this problem are presented in [52] and [7]).

- All of the above architectures are completely decentralized and do not keep any central state. As a result they may suffer from the lack of what Minsky would call a "B-brain" [47]. They can get stuck in loops or deadlock situations (i.e. keep activating the same actions even though they have proven not to result in any change of state).

- Most of the above architectures (apart from [14]) have a narrow-minded view of the relationship between perception and action. For example, few architectures support active or goal-driven perception, taking actions to obtain different or more sensor data, etc.

## 6.2 Learning from Experience

### 6.2.1 The Problem

The previous section discussed architectures for adaptive autonomous agents that focus on the problem of action selection. Almost all of these architectures neglect the issue of learning from experience (except for [35] [38]). This means that agents built using these architectures are only adaptive in a very restricted sense: they are able to deal with unexpected situations (opportunities, contingencies). However, these agents do not learn from environment feedback. They do not become better at achieving their goals with experience.

A second category of agent architectures that have been proposed has focussed on how the behavior (the action selection) of an agent can improve over time. Learning from experience is a necessity for any agent that has to demonstrate robust, autonomous behavior over long periods of time. First, this is the case because it is very hard to program an agent. It has practically proven impossible to correctly handcode a complex agent or to come up with a correct specification of its behavior and of the environment. Second, components of the agent may break down or its environment may change in a permanent way, which may require run-time "reprogramming". Adaptive behavior cannot be viewed as a final, static point. True adaptive behavior is an inherently dynamic, continuous process. It is in the spirit of the field of artificial life to view adaptive behavior as an emergent property of the long-term interaction and feedback process between an agent and its environment.

The problem of learning from experience can be defined as follows. Given an agent with (i) a set of actions or competence modules, (ii) certain sensor data and (iii) multiple (time-varying) goals, how can that agent improve its action selection behavior based on experience? How can the agent incorporate the feedback it receives after taking an action in such a way that its action selection behavior improves? "Improvement" typically means that the agent becomes more successful at fulfilling its goals or needs. Depending on the nature of the agent's goals, this may mean different things. In the case of an "attainment goal" or "end-goal" this would mean that the average time or average number of actions required (or any other measure of cost) to achieve the goal decreases over time. In the case of a reinforcement maximization type of goal, this could mean that the average positive reinforcement received over a fixed length time interval increases with experience.

No matter what the type of goals are it can deal with, any model for learning in an autonomous agent has to fulfill the following desiderata:

- The learning has to be incremental: the agent should learn with every experience. There cannot be a separate learning and performance phase.
- The learning should be biased towards learning knowledge which is relevant to the goals. In complex, realistic environments an agent cannot afford to learn every fact that can possibly be learned.
- The learning model should be able to cope with noise, probabilistic environments, faulty sensors, etc.
- The learning should be unsupervised. The agent has to learn mostly autonomously.
- Preferably, the learning model makes it possible to give the agent some initial built-in knowledge (so that it does not have to learn everything from scratch, in particular in those situations where prior knowledge is easily available).

There are three subproblems that have to be dealt with when designing an architecture for a learning agent:

1. What is the action selection mechanism adopted?
2. How does the system learn? How does it create "hypotheses" to be tested? And how does it decide which of these hypotheses are worth keeping or using to determine the behavior of the agent?
3. How does the agent decide when to "exploit" versus when to "explore"? How does it decide whether to activate whatever it believes is the most optimal action for the current situation versus whether to try a suboptimal action so as to learn and possible find a better way of doing things? That is, what is a good experimentation strategy for an agent?

Notice that with respect of the first of these problems, the learning architectures proposed often adopt a naive and limited view. Often the set of goals dealt with is very simple and the goals are fixed over time. Some more detailed problems come up when solving the above problems. For example, every learning architecture has to deal with the problem of credit assignment: which of the previously activated actions gets (partial) credit for a certain (desirable/undesirable) result happening?

How can we evaluate and compare different proposals for learning from experience? As with the problem of action selection, comparing proposals is hard to do in the general case. The problem of learning from experience is ill-defined unless one specifies what the particular characteristics of the environment, agent and task are. Therefore it only makes sense to compare proposals with respect to a particular class of problems. For example, an agent with a lot of memory might be better off using a memory-intensive learning method, rather than doing a lot of generalization to come up with a concise representation of what it has learned. In some environments initial knowledge is easily available, which means that it is desirable for the agent to be partially programmable (as opposed to learning from scratch). Depending on the environment and agent at hand, the role of learning may be very different. In an environment that is very predictable and that changes at a slower pace than the agent's lifetime, there is less of a need for learning during the agent's lifetime. Instead, some sort of evolution-based learning at the species level might be able to deal with the long term adaptation required [30]. Todd and Wilson [64] and [32] present some first steps towards a taxonomy of environments and agents that may make comparisons more meaningful.

### 6.2.2 Progress Made

All of the architectures that have been proposed in the literature assume that the agent has a set of primitive actions or competence modules. They concentrate on learning the arbitration network among these different actions or modules, i.e. the agent attempts to learn when certain action(s) should be activated (when an action should get control over the actuators). Some of the architectures proposed allow for learning of new "composite" actions or composite competence modules [41] [18]. They allow the agent to independently learn composite modules as well as the arbitration network for these composite modules.

The different architectures proposed can be grouped in three classes: reinforcement learning systems, classifier systems and model learners. The second class of architectures is really a special case of the first. However, since a lot of research has been performed in classifier systems, and since this research is not typically discussed from a reinforcement learning point of view, we will discuss the two classes separately. Both classes define the learning problem as follows: given a set of actions, given a reward signal, learn a mapping of situations to actions (called an "action policy") so that an agent following that policy maximizes the accumulated (discounted) reward it receives over time. In the case of a model learning architecture, the agent learns a model of how actions affect the environment (how actions map situations into other situations). Independent of this, the agent learns (or infers) what the importance or value of taking certain actions in certain situations is. Interesting combinations of these three types of architectures exist. For example, some systems combine learning of an action policy with learning of a model [8] [61].

As is the case with action selection models, many of the architectures proposed have been inspired by theories of animal learning. In contrast with the former, however, it is not so much the ethologist school of animal behavior studies, but rather comparative psychology and behaviorism, in particular theories of reinforcement learning and operant conditioning, that have been an inspiration for the computational models proposed.

*Reinforcement Learning.*

The idea of reinforcement learning [61] [62] [26] is the following. Given an agent with (i) a set of actions it can engage in, (ii) a set of situations it can find itself in, and (iii) a scalar reward signal that is received when the agent does something; learn an action policy, or a mapping from situations to actions, so that an agent which follows that action selection policy maximizes the cumulative discounted reward it receives over time.

Q-learning [68] is a particularly popular reinforcement learning strategy. In Q-learning the agent tries to learn for every situation-action pair, what the "value" is of taking that action in that situation. More specifically, the algorithm learns a two-dimensional matrix that stores a value for every possible combination of a situation and an action. At initialization, all values are set to some initial value. The goal of the system is to update these values so that they converge towards the "maximum cumulative discounted reward that can be expected when taking that action in that situation This means, the maximum cumulative reward that the agent can expect to receive in the future (from now on) if it takes that particular action in that situation (i.e. the immediate reward it receives plus the reward it will receive for taking the best future actions after this one). The reward is "discounted" with respect to the future so that rewards expected in the near future count for more than rewards expected further down the road. The different subproblems listed above are dealt with by reinforcement learning systems in the following way:

1. Action selection mechanism.
   At any moment, the agent always finds itself in some particular situation. Given that situation, it chooses the action that has the maximum value (maximum cumulative discounted reward)

2. Learning method.
   When the agent performs an action, it may receive some reward (possibly zero). It then updates the value of the situation-action pair it just "exploited". In particular, it increases or decreases the value of that situation-action pair so as to better reflect the actual reward it received plus the maximum reward it can expect in the new situation it finds itself in.

3. Exploration Strategy.
   In a certain percentage of situations, the agent does not choose the action that maximizes reward, but instead in performs a random action, so as to gather more data or evidence about possibly interesting alternative paths.

One of the attractive features of reinforcement learning is its formal foundation. It can be proven that under certain conditions (e.g. an infinite number of trials and a Markovian environment), the agent will converge towards the optimal action selection policy. Unfortunately these conditions are seldom attainable in real, complex situations. Disadvantages of reinforcement learning algorithms are (i) that they do not deal with time-varying goals (the action policy learned is for a fixed set of goals), (ii) if the goals change, they have to relearn everything from scratch ([27] attempts to overcome this problem), (iii) for realistic applications, the size of the state space (or the number of situation-action pairs) is so large that learning takes too much time to be practical (as a result, researchers have started developing algorithms that can generalize over the state space [41] [31]), (iv) learning only happens "at the fringe" of the state space (only when a reward is received can the system start learning about the sequence of actions leading to that reward), as a result it takes a lot of time to learn long action sequences ([61] attempts to deal with this problem), (v) the model assumes that the agent knows at all times which situation it is in (given faulty sensors or hidden states this is difficult) ([71] addresses this particular problem), (vi) it is hard to build in initial knowledge into this type of architecture and finally, (vii) the model cannot learn when multiple actions are taken in parallel[5].

[5] In theory, reinforcement learning can deal with parallel actions by adopting a row in the

*Classifier Systems.*

A second category of architectures for learning agents are based on classifier systems [20]. In particular Wilson [69] and Booker [8] have studied how classifier systems can be used to build adaptive autonomous agents. These architectures can be viewed a special case of reinforcement learning systems. That is, again, the agent attempts to learn how it can optimize the reward it receives for taking certain actions in certain situations. The idea here is that an agent has a set of rules, called "classifiers", and some data about every rule's performance. At the least the system keeps a "strength" for every rule that represents the value of that rule (how "good" it is). The three subproblems of a learning architecture are dealt with in the following way:

1. Action selection mechanism.
   Given a certain situation, which includes some external state (or sensor data) and may include an internal state, the condition list of some classifiers will match the current situation. Of all the matching classifiers, the agent picks one or more classifiers proportional to their strength. The actions proposed by those classifiers are executed (this may involve changing the internal state).

2. Learning method.
   Whenever some classifiers are executed, they give some of their strength to the classifiers that "set the stage", i.e. the classifiers that were just active at the previous timestep. This is called the "bucket-brigade algorithm" and is designed to deal with the problem of credit assignment. Whenever the agent executes some actions, it may receive some reward. If this is the case, then the reward will increase the strength of all the classifiers that were just activated (as well as all those that were not activated, but that suggested the same action). This scheme ensures that classifiers that contribute to a reward being received will over time have higher strengths than those that don't and will thus be activated more often.

3. Exploration strategy.
   The number of classifiers is fixed. Every once in a while, the agent removes those classifiers that have low strengths and replaces them by mutations and recombinations (crossovers) of successful ones. This way, the agent keeps exploring and evaluating different ways of doing things, while keeping "good" solutions around.

One of the interesting aspects of agents based on classifier systems is that they use a more sophisticated experimentation strategy (the experimentation strategy of other reinforcement learning systems consists of picking a random other data point). The hypothesis underlying this strategy is that one can

matrix for every combination of actions that is executed in parallel. In practice, however, this would blow up the state space even more than is already the case.

find a better solution to a problem (e.g. more effective behavior) by making small changes to an existing good solution (existing successful behavior) or by recombining existing promising solutions. Another advantage of classifier systems over most other reinforcement learning systems is that they have a built-in generalization mechanism for generalizing over situations as well as actions, namely the "#" or "don't care" symbol. This makes it possible for classifier systems to sample parts of the state space at different levels of abstraction and as such to find the most abstract representation of a classifier that is useful for a particular problem the agent has. Unfortunately, classifier system agents share some of the limitations of other reinforcement learning agents, in particular the problem of time varying goals (i) and the problem of learning at the fringe (iv) mentioned above. In addition they may suffer from the problem that they do not keep track of everything that has been tried. A classifier system based agent may re-evaluate the same classifier over and over again. It may throw it out because its strength is low and then immediately create it again because it keeps no memory of what has been tried (on the other hand, the fact that the system "forgets" about non promising classifiers makes it more efficient at action selection time).

*Model Builders.*
A final class of agents that learn from experience actually learn a causal model of their actions, rather than a policy map [18] [38] [50]. Drescher's model, which was inspired by Piaget's theories of development in infants, is probably the most sophisticated example. The agent builds up a probabilistic model of what the effects are of taking an action in a certain situation. This causal model can then be used by some arbitration process to decide which action is the most relevant given a certain situation and a certain set of goals. Action selection and learning are much more decoupled: in fact, the learning component of one of these agents could be combined with a different action selection mechanism.

In most of these architectures (except for [61]), the agent does not learn a complete mapping from every possible situation-action pair to the new situation that will result from taking that action in that situation. Rather, the agent learns mappings from partial situations to partial situations. It maps those aspects of a situation "that matter" (those sensor data that are necessary and sufficient conditions) combined with an action, to aspects of the new, resulting situation "that matter" (in particular sensor readings that change when taking the action in the situations described by the conditions). Such a combination of (i) a set of conditions, (ii) a primitive (or composite) action and (iii) a set of expected results (and probabilities for these results), is called a schema [18], or a module or behavior [38]. Model learning architectures deal with the three subproblems defined above in the following way:

1. Action Selection Mechanism.
   Agents built using these architectures can deal with time varying, multiple,

explicit goals. Given a set of goals and intensities, they compute at runtime which of the modules or schemas learned is most relevant to achieving the goal as well as most reliable. There is a separate value assignment process that is decoupled from the learning process. Often this value assignment process favors modules that prove to be more reliable. It may even trade off reliability of a sequence of actions for length of a sequence of actions leading to the goals. Typically a spreading activation process [38] or simple marker propagation process [18] is used to assign these values given some goals and sensor data.

2. Learning Method.
Whenever a particular action is taken, the agent monitors what changes happen in the environment. It uses this information to learn correlations between particular conditions-action pairs and certain results. After an action is taken, all result lists (and their probabilities) of applicable modules/schemas (that have the same action as the one taken and a matching condition list) are updated to reflect the new example. Occasionally, the agent needs to spin off new schemas from existing ones so as to be able to represent conflicting or unreliable results. The agent is able to detect that more conditions need to be taken into account in a schema/module for certain results to become more reliable, which will force it to create versions of the module that have longer, slightly different conditions.

3. Exploration Strategy.
The exploration strategy used in these architectures varies. Drescher's system, while demonstrating sophisticated learning, has an extremely simple exploration strategy, namely a random one. His agent basically does not do anything else but learn by performing random experiments. Foner [19] discusses how Drescher's agent can be made to learn much faster and to learn more relevant knowledge by adopting a smarter experimentation strategy as well as a focus of attention mechanism. In Maes' architecture [38], the exploration strategy is also more goal-oriented: the agent biases its experimentation towards actions that show promise to contribute to the goals. In the same system, the amount of exploration versus exploitation is an emergent property of the action selection system (the more has been learned, the fewer experiments are performed).

One of the main advantages of model learners is that they can transfer behavior learned in one context to another context (e.g. another goal). Since the system builds up a model of how taking an action in a situation results in another situation, it can use this model as a road map for any particular set of goals. As such they do better in environments where goals (or relative importances of goals) may change over time. This also implies that they do not just learn at the fringe of the state space connected to the goals. They learn from every action, as opposed to only learning from actions that have proven to be directly related to the present goals. In addition, they make it much easier for the designer of an agent to incorporate background knowledge about the domain (e.g. in the form of a causal model of the effects of actions). The agent is still able to correct this knowledge if it proves to be incorrect. The disadvantage of this type of architecture is that they may take more time to select an action, because there is no direct mapping of situations to "optimal" actions.

### 6.2.3 Open Problems

As is the case with action selection models, a lot of problems with modeling learning from experience remain unsolved. The following problems apply to all three of the above mentioned learning approaches:

- Scaling to larger (more realistic) problems is typically a problem for any of these learning algorithms. The computational complexity of all of the learning systems discussed is too big to be practically useful to build complex agents that solve real problems.
- One reason this is the case is that very few algorithms have incorporated interesting attention mechanisms. For example, Foner [19] demonstrates that incorporating attention mechanisms such as spatial locality can improve the tractability of learning from experience in a significant way. Most algorithms discussed above only use the temporal locality heuristic, i.e. effects are assumed to be perceived soon after the actions that caused them.
- Most of the algorithms proposed are bad at generalizing over sensor data. First, the sensor data are only represented at one level of granularity, as opposed to more coarse and finer levels. Second, none of the algorithms proposed exploit the structure and similarity present in many sensor data (e.g. one could exploit the fact that different cells of a retina are adjacent or that the different cells of the retina are affected in similar ways by certain actions).
- More work can be done in the domain of exploration strategies. Most existing algorithms employ the most simple strategy possible: the agent experiments a certain percentage of its time, no matter how urgent its needs or motivations may be, no matter how interesting the opportunities are that present themselves, etc. The agent also picks the experiment to perform in a random way, as opposed to using certain heuristics such as (i) trying actions that have not been tried for a while, (ii) trying actions that have shown promise recently, etc.
- There is a lack of interesting models of how learning and perception interact. The model of perception present in most architectures is narrow-

minded. The set of sensor data that the agent tries to correlate with its actions is taken as a given. The system does not couple learning about actions with learning about perception. It does not learn what to pay attention to or learn that more features should be paid attention to. Ideally, an agent would create new features and categories to perceive the environment based on whatever categories its goals and environment require (e.g. kittens that grow up in an environment with only vertical edges, do not develop detectors for horizontal edges).

- There is a lack of sophisticated models of how action selection and learning interact. In particular, all current algorithms assume that the set of primitive actions the agent learns about is a given. As is the case with sensor data, it would make more sense if the set of primitive actions is learned on the basis of what discretization or what subdivision of the continuous space of possible actions is appropriate for the environment and the goals at hand.

- We need to understand better what the role of learning is and how it interacts with other adaptive phenomena like cultural learning and adaptation through evolution (see [6]). We need to better understand what "building blocks" evolution could provide that could facilitate learning (e.g. provide a built-in bias for learning, or built-in specialized structures, etc).

- Finally, most of the approaches taken have been inspired by behaviorism and comparative psychology, rather than ethology. A lot could be learned by taking a more ethologically inspired approach to learning. For example, ethologists have shown that animals have built-in sensitive periods for learning particular competences. These periods tend to coincide with the situations in their lives that are optimal for picking up the competence to be learned, and as such reduce the complexity of the learning task.

# 7 Conclusions

Autonomous agent research represents an exciting new approach to the study of intelligence. So far, this new approach has demonstrated several "proofs of concept". In particular, encouraging successes have been reported in the area of mobile robots as well as software agents. Several prototypes have been built that have solved a real task that was previously not solvable or that was only solvable by means of a more costly and less effective solution. The approach has definitely had an impact on the course of artificial intelligence, which can be witnessed by the explosion of publications and research projects in the area.

Nevertheless, some problems are apparent that require novel ideas and better solutions. The main problem identified is that of scaling the approach to larger, more complicated systems. The tools and techniques proposed do not provide sufficient support to design or hand-build a complex agent with many different goals. The learning techniques proposed have computational complexities that make the automated development of an adaptive agent an intractable problem (in realistic time).

In addition, in order for the approach to be more founded, more fundamental research has to be undertaken. We need to understand the classes of problems agents have to deal with, so that it becomes possible to critically compare particular architectures and proposals. For example, many different models of action selection have been proposed, but unless we understand the problem of action selection better, we do not have any grounds to compare the different proposals.

Aside from better evaluation criteria, we need a better understanding of the underlying principles. In particular, it is important to understand the mechanisms and limitations of emergent behavior. How can a globally desired structure or functionality be designed on the basis of interactions between many simple modules? What are the conditions and limitations under which the emergent structure is stable, and so on. Some first steps towards a theory of emergent functionality have been proposed, using tools from complex dynamics [58] [28] [5]. However, so far the proposed theories have only been applicable to very simple toy examples.

There is tension inherent in the agent approach that is as of now unresolved. Research in autonomous agents has adopted very task-driven, pragmatic solutions. As a result, the agents built using this approach end up looking more like "a bag of hacks and tricks", than an embodiment of a set of more general laws and principles. Does this mean that the field will evolve into a (systems) engineering discipline, or will we find a path towards becoming a more scientific discipline?

## Acknowledgements

Bruce Blumberg, Leonard Foner, Chris Langton, Yezdi Lashkari, Maja Mataric and Stewart Wilson provided valuable comments on an earlier draft of this paper. Chris Langton provided the necessary encouragement and demonstrated a lot of patience.

## References

[1] Agre P.E., The Dynamic structure of Everyday Life, Cambridge University Press, 1991.

[2] Ballard D.H., Reference Frames for Animate Vision, Proceedings of IJCAI-89 conference, Detroit, 1989.

[3] Bates, J., Loyall B. & Reilly W., Broad Agents, Proceedings of the AAAI Spring Symposium on Integrated Intelligent Architectures, Stanford, CA. Available in SIGART Bulletin, Vol. 2 (4), pp 38-40, 1991.

[4] Beer R., Chiel H. & Sterling L., A Biological Perspective on Autonomous Agent Design, In: Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, edited by P. Maes, MIT Press/Bradford Books, 1990.

[5] Beer R., A Dynamical Systems Perspective on Autonomous Agents, Technical Report CES-92-11, Department of Computer Engineering and Science, Case Western Reserve University, 1992.

[6] Belew R., Evolution, Learning and Culture: Computational Metaphors for Adaptive Algorithms, UCSD Computer Science and Engineering Department, CSE Technical Report #CS89-156, 1989.

[7] Blumberg B., Action-Selection in Hamsterdam: Lessons from Ethology, Submitted to: the Third International Conference on the Simulation of Adaptive Behavior, Brighton, August 1994.

[8] Booker L., Classifier Systems that Learn Internal World Models, Machine Learning Journal, Volume 1, Number 2,3, 1988.

[9] Braitenberg V., Vehicles: Experiments in Synthetic Psychology, MIT Press/Bradford Books, 1984.

[10] Brooks R.A., A Robust Layered Control System for a Mobile Robot, IEEE Journal of Robotics and Automation, RA-2, April 1986.

[11] Brooks R.A., Intelligence without Reason, Computers and Thought lecture, Proceedings of IJCAI-91, Sidney, Australia, 1991.

[12] Brooks R.A., Challenges for Complete Creature Architectures, In: From Animals to Animats, Proceedings of the First International Conference on the Simulation of Adaptive Behavior, edited by Meyer J.-A. & Wilson S.W., MIT Press/Bradford Books, 1991.

[13] Brooks R.A., Artificial Life and Real Robots, In: Toward a Practice of Autonomous Systems, Proceedings of the First European Conference on Artificial Life, edited by Varela, F. & Bourgine P., MIT Press/Bradford Books, 1992.

[14] Chapman D., Vision, Instruction and Action, MIT Press, 1992.

[15] Connell J., Minimalist Mobile Robotics, Academic Press, 1990.

[16] Deneubourg J.L., Goss S., Franks N., Sendova-Franks A., Detrain C., Chretien L., The Dynamics of Collective Sorting: Robot-like Ants and Ant-like Robots, In: From Animals to Animats, Proceedings of the First International Conference on the Simulation of Adaptive Behavior, edited by Meyer J.-A. & Wilson S.W., MIT Press/Bradford Books, 1991.

[17] Deneubourg J.L., Theraulaz G. & Beckers R., Swarm-Made Architectures, In: Toward a Practice of Autonomous Systems, Proceedings of the First European Conference on Artificial Life, edited by F.J. Varela & P. Bourgine, MIT Press/Bradford Books, 1992.

[18] Drescher G.L., Made-Up Minds: A Constructivist Approach to Artificial Intelligence, MIT Press, 1991.

[19] Foner L., Paying Attention to What's Important: Using Focus of Attention to Improve Unsupervised Learning. Submitted to: the Third International Conference on the Simulation of Adaptive Behavior, Brighton, August 1994.

[20] Holland J.H., Escaping Brittleness: the Possibilities of General-Purpose Learning Algorithms applied to Parallel Rule-Based Systems, In: Machine Learning, an Artificial Intelligence Approach, Volume II, edited by R.S. Michalski, J.G. Carbonell & T.M. Mitchell, Morgan Kaufmann, 1986.

[21] Holland J.H., The Optimal Allocation of Trials (Chapter 5), in: Adaption in Natural and Artificial Systems, MIT Press/Bradford Books, 1992.

[22] Horswill I., Characterizing Adaptation by Constraint, In: Toward a Practice of Autonomous Systems, Proceedings of the First European Conference on Artificial Life, edited by F.J. Varela & P. Bourgine, MIT Press/Bradford Books, 1992.

[23] Horswill I., Specialization of Perceptual Processes, PhD thesis, AI Laboratory, MIT, 1993.

[24] Kaelbling L.P. & Rosenschein S., Action and Planning in Embedded Agents, In: Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, edited by P. Maes, MIT Press/Bradford Books, 1990.

[25] Kaelbling L.P., An Adaptable Mobile Robot, In: Toward a Practice of Autonomous Systems, Proceedings of the First European Conference on Artificial Life, edited by F.J. Varela & P. Bourgine, MIT Press/Bradford Books, 1992.

[26] Kaelbling L.P., Learning in Embedded Systems, MIT Press, 1993.

[27] Kaelbling L.P., Learning to Achieve Goals, In: Proceedings of IJCAI-93, the Thirteenth International Joint Conference on Artificial Intelligence, Morgan Kaufman, 1993.

[28] Kiss G., Autonomous Agents, AI and Chaos Theory, In: From Animals to Animats, Proceedings of the First International Conference on the Simulation of Adaptive Behavior, edited by Meyer J.-A. & Wilson S.W., MIT Press/Bradford Books, 1991.

[29] Kleinrock L. & Nilsson A., On Optimal Scheduling Algorithms for Time-Shared Systems, Journal of the ACM, 28, 3, July 1981.

[30] Koza J.R., Evolution and Co-evolution of Computer Programs to Control Independently-Acting Agents, In: From Animals to Animats, Proceedings of the First International Conference on the Simulation of Adaptive Behavior, edited by Meyer J.-A. & Wilson S.W., MIT Press/Bradford Books, 1991.

[31] Lin L.-J., Reinforcement Learning for Robots using Neural Networks, PhD thesis, Carnegie Mellon University, School of Computer Science, 1992.

[32] Littman M., An Optimization-Based Categorization of Reinforcement Learning Environments. In: From Animals to Animats 2, Proceedings of the Second International Conference on Simulation of Adaptive Behavior, edited by Meyer J.-A., Roitblat H.L. & Wilson S.W., MIT Press/Bradford Books, 1993.

[33] Maes P., Situated Agents Can Have Goals, In: Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, edited by P. Maes, MIT Press/Bradford Books, 1990.

[34] Maes, P., Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, MIT Press/Bradford Books, 1990.

[35] Maes P. & Brooks R.A., Learning to Coordinate Behaviors, Proceedings of AAAI-90, Boston, 1990.

[36] Maes P., Adaptive Action Selection, Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society, Lawrence Erlbaum Associates, 1991.

[37] Maes, P., A Bottom-Up Mechanism for Behavior-Selection in an Artificial Creature, In: From Animals to Animats, Proceedings of the First International Conference on the Simulation of Adaptive Behavior, edited by Meyer J.-A. & Wilson S.-W., MIT Press/Bradford Books, 1991.

[38] Maes P., Learning Behavior Networks from Experience, In: Toward a Practice of Autonomous Systems, Proceedings of the First European Conference on Artificial Life, edited by F.J. Varela & P. Bourgine, MIT Press/Bradford Books, 1992.

[39] Maes P., 1993, ALIVE: An Artificial Life Interactive Video Environment, Visual Proceedings of the Siggraph-93 Conference, ACM Press, 1993.

[40] Maes P. & Kozierok R., Learning Interface Agents, Proceedings of AAAI-93, the Eleventh National Conference on Artificial Intelligence, MIT Press, 1993.

[41] Mahadevan S. & Connell, J., Automatic Programming of Behavior-Based Robots using Reinforcement Learning, In: Proceedings of the Ninth National Conference on Artificial Intelligence, MIT Press, 1991.

[42] Malone T.W., Fikes R.E., Grant K.R., & Howard M.T., Enterprise: A Market-like Task Scheduler for Distributed Computing Environments, In: The Ecology of Computation, edited by B. Huberman, North-Holland, 1988.

[43] Mataric M.J., Behavioral Synergy without Explicit Integration, In: Special Issue of SIGART on Integrated Cognitive Architectures, Volume 2, Number 4, 1991.

[44] Mataric, M.J., Integration of Representation Into Goal-Driven Behavior-Based Robots, *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 3, June 1992, 304-312.

[45] McFarland D., What it Means for a Robot Behavior to be Adaptive, In: From Animals to Animats, Proceedings of the First International Conference on the Simulation of Adaptive Behavior, edited by Meyer J.-A. & Wilson S.-W., MIT Press/Bradford Books, 1991.

[46] Meyer J.-A. & Guillot A., Simulation of Adaptive Behavior in Animats: Review and Prospects, In: From Animals to Animats, Proceedings of the First International Conference on the Simulation of Adaptive Behavior, edited by Meyer J.-A. & Wilson S.-W., MIT Press/Bradford Books 1991.

[47] Minsky M., The Society of Mind, Simon and Schuster, New York, 1986.

[48] Nilsson N., Shakey the Robot, SRI A.I. Center Technical Note 323, 1984.

[49] Nilsson N.J., Toward Agent Programs with Circuit Semantics, Department of Computer Science, Report number STAN-CS-92-1412, Stanford University, January 1992.

[50] Payton D.W., Keirsey D., Krozel J. & Rosenblatt K., Do Whatever Works: A Robust Approach to Fault-Tolerant Autonomous Control, Journal of Applied Intelligence, Vol. 3, 1992.

[51] Resnick M., Beyond the Centralized Mindset: Explorations in Massively Parallel Microworlds, PhD Thesis, MIT Media-Laboratory Epistemology and Learning Group, 1992.

[52] Rosenblatt J. & Payton D., A Fine-Grained Alternative to the Subsumption Architecture for Mobile Robot Control, In: Proceedings of the International Joint Conference on Neural Networks, IJCNN, Washington DC, June 1989.

[53] Sheth B. & Maes P., Evolving Agents for Personalized Information Filtering, Proceedings of the IEEE Conference on Artificial Intelligence for Applications, IEEE Press, 1993.

[54] Shrager J. & Callanan M., Active Language in the Collaborative Development of Cooking Skill, Proceedings of the Cognitive Science Conference, Lawrence Erlbaum, 1991.

[55] Simon H., The Sciences of the Artificial, MIT Press, 1969.

[56] Steels L., Cooperation between Distributed Agents through Self-Organization, In: Decentralized AI, edited by Y. Demazeau & J.-P. Muller, Elsevier, North-Holland, 1990.

[57] Steels L., Exploiting Analogical Representations, In: Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, edited by P. Maes, MIT Press/Bradford Books, 1990.

[58] Steels L., Towards a Theory of Emergent Functionality, In: From Animals to Animats, Proceedings of the First International Conference on the Simulation of Adaptive Behavior, edited by Meyer J.-A. & Wilson S.W., MIT Press/Bradford Books, 1991.

[59] Suchman L.A., Plans and Situated Actions: The Problem of Human-Machine Communication, Cambridge University Press, 1987.

[60] Sullivan J.W. & Tyler S.W. (editors), Intelligent User Interfaces, ACM Press, 1991.

[61] Sutton R.S., Integrated Architectures for Learning, Planning and Reacting based on Approximating Dynamic Programming, Proceedings of the Seventh International Conference in Machine Learning, 1990.

[62] Sutton R. S., Reinforcement Learning Architectures for Animats, In: From Animals to Animats, Proceedings of the First International Conference on the Simulation of Adaptive Behavior, edited by Meyer J.-A. & Wilson S.W., MIT Press/Bradford Books, 1991.

[63] Toates F. & Jensen P., Ethological and Psychological Models of Motivation-Towards a Synthesis, In: From Animals to Animats, Proceedings of the First International Conference on the Simulation of Adaptive Behavior, edited by Meyer J.-A. & Wilson S.W., MIT Press/Bradford Books, 1991.

[64] Todd P. & Wilson S., Environment Structure and Adaptive Behavior from the Ground Up. In: From Animals to Animats 2, Proceedings of the Second International Conference on Simulation of Adaptive Behavior, edited by Meyer J.-A., Roitblat H.L. & Wilson S.W., MIT Press/Bradford Books, 1993.

[65] Travers M., Animal Construction Kits, In: Artificial Life, edited by Langton C., Addison Wesley, 1989.

[66] Tyrrell T., Defining the Action Selection Problem, In: Proceedings of the 14th Conference of the Cognitive Science Society, 1992.

[67] Tyrrell T., Computational Mechanisms for Action Selection, PhD Thesis, Centre for Cognitive Science, University of Edinburgh, 1993.

[68] Watkins C., Learning from Delayed Rewards, PhD Thesis, King's College, Cambridge, 1989.

[69] Wilson S.W., Knowledge Growth in an Artificial Animal, In: Proceedings of the First International Conference on Genetic Algorithms and their Applications, edited by Greffenstette, Lawrence Erlbaum Associates, 1985.

[70] Wilson S.W., The Animat Path to AI, In: From Animals to Animats, Proceedings of the First International Conference on the Simulation of Adaptive Behavior, edited by Meyer J.-A. & Wilson S.W., MIT Press/Bradford Books, 1991.

[71] Whitehead S. & Ballard D., Active Perception and Reinforcement Learning, In: Proceedings of the Seventh International Conference on Machine Learning, Austin, Texas, 1990.