Cross-Paradigm Analysis of Autonomous Agent Architecture

Joanna Bryson
Division of Informatics
The University of Edinburgh
2 Buccleuch Place
Edinburgh, EH8 9NG
United Kingdom

+44 131 650 4454

Running Title: Cross-Paradigm Analysis of Agent Architecture

Abstract

Autonomous agent architectures are design methodologies — collections of knowledge and strategies which are applied to the problem of creating situated intelligence. In this article, we attempt to integrate this knowledge across several architectural traditions. We pay particular attention to features which have tended to be selected under the pressure of extensive use in real-world systems. We conclude that the following strategies provide significant assistance in the design of autonomous intelligent control:

- *Modularity*, which simplifies both design and control,
- *Hierarchically Organized Action Selection*, which allows for focusing attention and providing prioritization when different modules conflict, and
- a parallel *Environment Monitoring System* which allows a system to be responsive and opportunistic by allowing attention to shift and priorities to be reevaluated.

We offer a review of four architectural paradigms: behavior-based AI; two and three layered systems; belief, desire and intention architectures (particularly PRS); and Soar / ACT-R. We document trends within each of these communities towards establishing the components above, arguing that this convergent evolution is strong evidence for the components' utility. We then use this information to recommend specific strategies for researchers working under each paradigm to further exploit the knowledge and experience of the field as a whole.

1 Introduction

1.1 Approach

Agent architectures are design methodologies. The assortment of architectures our community uses reflects our collective knowledge about what methodological devices are useful when trying to build an intelligence. We consider this perspective, derived from Maes (1991a) and Wooldridge and Jennings (1995), to be significantly more useful than thinking of an architecture as a uniform skeletal structure specified by a particular program. The definition of an agent architecture as a collection of knowledge and methods provides a better understanding of how a single architecture can evolve (e.g. Laird and Rosenbloom, 1994; Myers, 1996) or two architectures can be combined (e.g. Bryson, 1999a).

The design knowledge expressed in agent architectures is of two types: knowledge derived by reasoning, and knowledge derived by experience. Knowledge derived by reasoning is often explicit in the early papers on an architecture: these ideas can be viewed as hypotheses, and the intelligences implemented under the architecture as their evidence. Knowledge derived by experience may be more subtle: though sometimes recognized and reported explicitly, it may be hidden in the skill sets of a group of developers. Worse yet, it may be buried in an unpublished record of failed projects or missed deadlines. Nevertheless, we would like to premise this paper on the idea that facts about building intelligence are likely to be found in the history and progress of agent architectures. That is, we assume that architectures will tend to include the attributes which have proven useful over time and experience.

Unfortunately, as with most selective processes, it is not always a simple matter to determine for any particular expressed attribute whether it has itself proven useful. A useless feature may be closely associated with other, very useful attributes, and consequently be propagated through the community as part of a well-known, or well-established architecture. Similarly, dominating architectures may lack particular useful elements, but still survive due to a combination of sufficient useful resources and sufficient communal support. For these reasons alone one cannot expect any particular architecture to serve as an ultimate authority on design methodology, even if one ignores the arguments of niche specificity for various architectures. But we do assume that architectural trends can be used as evidence for the utility of a particular design approach.

Identifying the design advantage behind such trends can be useful, because it allows the research community to further develop and exploit the new methodology. This is truer not only within the particular architecture or architectural paradigm in which the trend emerged, but can also benefit the autonomous control community in general. To the extent that all architectures face the same problems of supporting the design of intelligence, any development effort may benefit from emphasizing strategies that have proven useful. Many architectures have a larger number of features than their communities typically utilize. In other words, many architectures are under-specified as a design methodology. Consequently, even established design efforts may be able to exploit new knowledge of design strategy without changing their architectural software tools. They may be able to make simple reorganizations or additions to their established design processes.

In this article, we will demonstrate this approach for evaluating and enhancing agent architectures. We survey the dominant paradigms of agent architecture technology: behavior-based design; two- and three-layer architectures; PRS and the belief, desire and intention architectures; and Soar and ACT-R. We begin by looking at some of the historic concerns about architectural approach that have shaped and differentiated these communities. We then review each paradigm and the systematic changes which have taken place within it over the last 15 years. We conclude with a discussion of these architectures in terms of the lessons derived from that review, making recommendations for the next stages of development for each paradigm.

Thesis

To improve the overall coherence of this article, we begin with the results of our research on these and other architectures. We have concluded that there are several necessary architectural attributes for producing an agent that is both reactive and capable of complex tasks. One is an explicit means for ordering action selection, in particular a mechanism exploiting hierarchical and sequential structuring. Such a system allows an agent with a large skill set to focus attention and select appropriate actions quickly. This has been a contentious issue in agent architectures, and this controversy is reviewed below. The utility of hierarchical control has been obscured by the fact it is not itself sufficient. The other necessary components include a parallel environment monitoring system for agents in dynamic environments, and modularity, which seems to benefit all architectures.

Modularity substantially simplifies the design process by substantially simplifying the individual components to be built. In this article, we will define modularity to be the decomposition of an agent's intelligence, or some part of its intelligence, into a number of smaller, relatively autonomous units. However, we do not mean to necessarily imply the fully encapsulated modules of Fodor (1983), where the state and functionality of one module are strictly unavailable to others. The most useful form of modularity seems to be decomposed along the lines of ability, with the module formed of the perception and action routines necessary for that ability, along with their required or associated state.

Fully modular architectures can create new design difficulties. If sequential and hierarchical control are avoided, then specifying priority and action selection between the interacting modules becomes difficult. However, an architecture that does allow for a centralized control system to focus attention and select appropriate actions may fail to notice dangers or opportunities that present themselves unexpectedly. Agents existing in dynamic environments must have architectural support for monitoring the environment for significant changes, in order for the complete agent to remain responsive. This may either be a part of the main action-selection system, or a separate system with priority over primary action selection.

2 Background

2.1 The Traditional Approach

A traditional architecture for both psychology and artificial intelligence is shown in Figure 1. This architecture indicates that the problems of intelligence are to transform perception into a useful mental representation R; apply a cognitive process f to R to create R', a representation of desired actions; and transform R' into the necessary motor or neural effects. This model has lead many intelligence researchers to feel free to concentrate on only a single aspect of this theory of intelligence, the process between the two transformations, as this has been considered the key element of intelligence.

— Figure 1 about here. —

This model (in Figure 1) may seem sufficiently general as to be both necessarily correct and uninformative, but in fact it makes a number of assumptions known to be wrong. First, it assumes that both perception and action can be separated successfully from cognitive process. However, perception is known to be guided by expectations and context — many perceptual experiences cannot be otherwise explained (e.g. Neely, 1991; McGurk and MacDonald, 1976). Further, brain lesion studies on limb control have shown that many actions require constant perceptual feedback for control, but do not seem to require cognitive contribution, even for their initiation (e.g. Matheson, 1997; Bizzi et al., 1995).

A second problem with this architecture as a hypothesis of intelligence is that the separation of representation from cognitive process is not necessarily coherent. Many neural theories postulate that an assembly of neurons processes information from perception, from themselves and from each other (e.g. McClelland and Rumelhart, 1988; Port and van Gelder, 1995). This processing continues until a recognized configuration is settled. If that configuration involves reaching the critical activation to fire motor neurons, then there might be only one process running between the perception and the activity. If the levels of activation of the various neurons are taken as a representation, then the process is itself a continuous chain of re-representation. Notice that the concept of a "stopping point" in cognition is artificial — the provision of perceptual information and the processing activity itself is actually continuous for any dynamic agent. The activations of the motor system are incidental, not consummatory.

2.2 Hierarchical Theories of the Organization of Intelligent Behavior

The distinctions made in describing the basic traditional architecture and the role of intervening re-representation have lead to a number of controversies which have in turn influenced the various approaches to agent architectures. In particular, there has been considerable controversy over whether behavior is organized using privileged hierarchical and sequential structures, or whether its order emerges from a much more dynamic process. It has been argued (by e.g. Vereijken and Whiting, 1998; van Gelder, 1998; Hendriks-Jansen, 1996; Goldfield, 1995; Maes, 1991b) that hierarchical strategies of action selection necessarily lead to rigid, brittle systems incapable of reacting quickly and opportunistically to changes in the environment. On the other hand, hierarchical and sequential control are well-established programming techniques that demonstrably manage enormous complexity. This section and the following one discuss this history.

Initially it appears obvious that there must be some structure or plan to behavior. Such structures radically simplify the problem of choosing a next behavior by reducing the number of options that need to be evaluated in selecting the next act. The standard strategy for AI systems is to convert perceptions and intentions into a plan — a sequence or partial ordering of behaviors which will attain the current goal. The term "plan" in this context does not necessarily indicate intentionality, nor even the conventional sense of "planning". Rather, it refers to a plan as established course for action selection, a sort of blueprint for action. Whether this blueprint is a result of instinct, past experience or an immediate creative process is a separate, though related question.

Assuming a hierarchical model, some plans may be of different origins than others. A plan is considered hierarchical if its elements might in turn be plans. For example, if a dog is hungry, it might go to the kitchen, then rattle its bowl. Going to the kitchen would itself entail finding a path through the house, which involves moving through a series of locations. Moving between locations itself requires a series of motor actions. The theory of hierarchical control supposes that the mechanisms responsible for the fine muscle control involved in the dog's walking are not the same as those responsible for choosing its path to the kitchen, and these in turn are not necessarily the concern of the system that determined to move to the kitchen the first place. A plan is considered sequential to the extent that its elements deterministically follow each other in a fixed order, for example the order in which a dog's feet are raised and advanced while it is moving with a particular gait.

Hendriks-Jansen (1996) traces the hierarchical theory of behavior organization in animals and man to the ethologist McDougall (1923), who presented a theory of the hierarchy of instincts. Ethological theory during this period, however, was dominated by Lorenz, who "denied the existence of superimposed mechanisms controlling the elements of groups" instead believing that "the occurrence of a particular activity was only dependent on the external stimulation and on the threshold for release of that activity."

(Baerends 1976 p. 726 cited in Hendriks-Jansen 1996 pp. 233–234). This theory dominated until Lashley (1951) reintroduced the idea of hierarchical organization of behavior. Lashley supports hierarchy on the basis that there could be no other explanation for the speed of some action sequences, such as those involved in human speech or the motions of the fingers on a musical instrument. Neural processes are simply too slow to allow elements of such sequences to be independently triggered in response to one another. Lashley therefore proposes that all the elements of such a sequence must be simultaneously activated by a separate process — the definition of hierarchical organization.

Lashley's argument has been taken up and extended by Dawkins (1976), who further argues for hierarchical theories of control for reasons of parsimony. Dawkins argues that it is more likely that a complex action sequence useful in multiple situations should be evolved or learned a single time, and that it is also more efficient to store a single instance of such a skill. Dawkins' arguments and proposals anticipate many the techniques developed later for robust control in artificial intelligence (e.g. Brooks, 1986; Maes, 1991a).

From roughly the time of Lashley's analysis, hierarchical models have been dominant in attempts to model intelligence. Particularly notable are the models of Tinbergen (1951) and Hull (1943) in ethology, Chomsky (1957) in linguistics, and Newell and Simon (1972) in artificial intelligence and human problem solving. Mainstream psychology has been less concerned with creating specific models of behavior control, but generally assumes hierarchical organization as either an implicit or explicit consequence of goal directed or cognitive theories of behavior (Bruner, 1982). Staged theories of development and learning are also hierarchical when they described complex skills being composed of simpler, previously-developed ones (Piaget, 1954; Karmiloff-Smith, 1992; Greenfield, 1991).

Nevertheless, the problem of how this apparently hierarchically ordered behavior emerges from a brain or machine has been an ongoing issue, in psychology as well as AI. The well-known defenses of Lashley and Dawkins are still being revisited (Houghton and Hartley, 1995; Nelson, 1990). In response to a recent target article on imitation in apes which assumed hierarchical structure to behavior (Byrne and Russon, 1998), nearly a fifth of the commentaries chosen to appear with the article questioned the existence of hierarchical control (Vereijken and Whiting, 1998; Mac Aogáin, 1998; Jorion, 1998; Gardner and Heyes, 1998, of 21 commentaries). As Gardner and Heyes (1998) point out, "The mere fact that [one] can describe behavior in terms of goals and subgoals is not evidence that the behavior was executed under hierarchical control."

2.3 Fully Distributed Theories of Intelligence

The competing viewpoint, that responsive intelligence cannot possibly be governed by hierarchical control, has come from some of the practitioners of the dynamical hypothesis of cognition (Port and van Gelder, 1995; van Gelder, 1998). The theory of dynamical action expression suggests that complex dynamical or chaotic systems operate within the brain producing the next behavior not by selecting an element of a plan, but rather as an emergent consequence of many parallel processes (e.g. Minsky, 1985; McClelland and Rumelhart, 1988; Maes, 1991b; Brooks, 1991; Goldfield, 1995; Kelso, 1995; Hendriks-Jansen, 1996; van Gelder, 1998). Evidence supporting the older hypothesis of structured hierarchical behavior is seen to have been biased by the hierarchical and sequential nature of human explicit thought and language. In particular, because much theoretical work in psychology is conducted using computer models, theories may be biased towards the workings and languages of the serial processors of the machines available to most psychologists (Brooks, 1991).

This argument for the emergent view of behavior ordering has been developed by two related but to some extent antithetical perspectives. Some support comes from researchers presuming the modularity of intelligence. Currently this is much of the behavior-based AI community; earlier this century it was the behaviorists. Modularists dismissing structured control prefer the hypothesis that each modular unit should determine its own expression, using its own perceptive abilities. Several agent architectures demonstrating this viewpoint are discussed below (but see for reviews Maes, 1990a; Blumberg, 1996). The other source of support for emergence over explicit structure comes from researchers who favor dynamical systems theory. This field, arising from physics and mathematics, is related to chaos theory. In its most extreme form, dynamical systems theory resists thinking of any of the universe, let alone intelligence, as segmented or modular. The resistance to hierarchy and sequence is that such organization implies segmentation of process. In their view process is continuous (Goldfield, 1995; Bohm, 1980). In this article, we will refer to both of these paradigms as fully distributed theories of intelligence, since they share the attributes of decomposing traditional theories of intelligence, and explaining apparent coherence as emergent phenomena.

Of course, both of these fully distributed communities have been forced to address the issue of complex sequences of behavior. The behaviorists initially believed that all behavior sequences are learned as chains of conditioned responses. This hypothesis has been disproved. The behaviorists themselves showed both that reward fails to propagate over long sequences and that complex structures such as mazes are learned through latent mechanisms which require no external reward (see Adams, 1984, for a review and history). Further, neuroscience has shown that the fine coordination between different elements of behaviors such as speech and piano playing occur too rapidly to allow for triggering of one act by its predecessor (Lashley, 1951). Dynamical theories have been successful at describing rapid, integrated sets of behavior, provided these can be described as a part of a rhythmic whole. For example, swimming and walking can be explained and controlled through understood mathematical mechanisms providing there exists a pattern generator to keep the rhythm of the cycle (Beer, 1995; Goldfield, 1995; Reeves and Hallam, 1995). Such oscillatory pattern generators are basic neural structures found in all vertebrate life (Carlson, 1994). However, the structuring of more complicated, heterogeneous behaviors such as nest building, starting a car, or speaking a sentence, have not yet been successfully addressed (see Saltzman, 1995, for an example of a well analyzed near miss). Further, the significantly greater combinatorial complexity such heterogeneity opens up is often either ignored or dismissed (e.g. Bohm (1980, p.182) or Goldfield (1995, pp. 285–288)).

This is not to deny the appeal of the fully distributed hypotheses, and their successes in advancing our understanding of natural intelligence. One appeal of fully distributed theories of intelligence is that they better explain the fact that errors are made in sequencing of even familiar tasks (Norman and Shallice, 1986; Henson, 1996). Another is that they take into account and better explain the time course of behaviors that traditional architectures treat as discrete events, such as decision making (Townsend and Busemeyer, 1995). Perhaps most importantly from the perspective of agent architectures, fully distributed theories allow for new information to constantly influence the course of actions. Thus if the dog, described earlier, on its way to the kitchen happens to pass a dropped sandwich, the action “eat what’s here” should suddenly overtake “visit the kitchen.” Systems with this capacity to be responsive and opportunistic are described in the artificial intelligence literature as being *reactive*.

A fundamental strength of the fully distributed hypothesis is that it is necessarily correct, at least to some level. Assuming a materialist stance, intelligence is known to be based in the parallel operation of the body’s neural and endocrine systems, which are indeed highly distributed. On the other hand, it is nearly as well accepted that human and animal behavior can be described as hierarchically ordered (Dawkins, 1976; Greenfield, 1991; Byrne and Russon, 1998). The question is, how and when are these behaviors so organized? Is the order purely apparent, or does it emerge in the brain prior to the external expression of behavior? These questions have a mirror in the architectural choices of the AI researcher: should there be a dedicated and centralized action-selection mechanism, or would such a system be a bottle-neck for intelligence, resulting in a sluggish and unresponsive system? The latter is the claim of Maes (1991b), who has had substantial impact on the modular agent architecture paradigm. Consequently, her claim has been examined empirically as is documented in Section 3.1 below.

Other architectural trends (e.g. PRS, Georgeff and Lansky, 1987, Section 3.3 below) attempt to combine some centralized hierarchical control with a reactive component attendant at the decision points. This also falls under harsh criticism from some champions of the fully distributed approach:

> [In hybrid systems] Environmental contingencies play a part at the choice points and in the form of orienting feedback, but the part they play can be explained only in terms of the entities manipulated by the program, which of course takes the form of a temporal sequence of formally defined instructions.
>
> Nearly forty years of experience in AI have shown that such a control mechanism soon

> gets into trouble in the real world because of its lack of flexibility, the need to plan for all possible contingencies, the combinatorial explosion, the frame problem, and the problems of interfacing a formally defined planner, working with an internal representation of the world conceptualized as a task domain of objects, properties, and events, to effectors and receptors that need to deal with a noisy real world that clearly is not pre-registered into objects, properties, and events. (Hendriks-Jansen, 1996, page 245.)

Hendriks-Jansen (1996) is describing the architecture of Hull (1943) here, but his description fits a number of recent hybrid architectures. It is representative of the criticism of plan-like elements in an agent: he carefully covers both the alternative of having plans as pre-existing elements of an agent's intelligence and that of having plans constructed by a separate module. His evaluation of the lessons of AI is however based mostly on work performed before 1990, with selective exceptions in the early part of that decade. Section 3 of this article provides a considerable update to this review.

2.4 Newer Approaches

2.4.1 A Modular Architecture

All of these more recent architectures have been influenced by the successes of the reactive and behavior-based movements in AI of the late 1980s. As described earlier, behavior-based architectures assume that a useful and accurate way to model intelligence is to model behavioral skills independently of each other. The extreme view in this field is that intelligent behavior cannot emerge from thinking and planning, but rather that the appearance of planning and thinking will emerge from behaving intelligently in a complex world (Brooks, 1991).

Though traceable in philosophy at least as far back as Hume (1748), and in psychology as far back as Freud (1900), the notion of decomposing intelligence into semi-autonomous independent agencies was first popularized in AI by Minsky (1985). Minsky's model promotes the idea of multiple **agencies** specialized for particular tasks and containing specialized knowledge. Minsky proposes that the control of such units would be easier to evolve as a species or learn as an individual than a single monolithic system. He also argues that such a model better describes the diversity and inconsistency of human behavior.

Minsky's "agents of mind" are hierarchical and only semi-autonomous. For example, he postulates, a child might have separate agencies for directing behavior involving sleeping, eating and playing. These compete for control. When a victorious agent emerges, its subsidiary agencies in turn compete. Once playing is chosen, blocks compete with dolls and books; if blocks are chosen, building and knocking down compete during the block-playing episode. Meanwhile, the agency in charge of eating may overwhelm the agency in charge of playing, and coherent behavior may be interrupted in mid-stride as different agencies swap to take control.

The cost of theories that successfully explain the incoherence of human thought and activity is that they often fail to explain its coherence. Minsky addresses this by postulating a modular rather than a completely distributed system of thought. He explains coherent behavior as being the output of a single agency or suite of agents, and incoherence as a consequence of competing agencies. He also recognizes that there can be coherent transitions between apparently modular behaviors. To address this, he postulates another type of structure, the *k-line*. K-lines connect modules associated in time, space, or as parts of the same entity. He also posits fairly traditional elements of knowledge representation, frames and knowledge hierarchies, for maintaining databases of knowledge used by the various agents.

2.4.2 A Reactive Architecture

Brooks, quoted above, took modularity to a greater extreme when he established the behavior-based movement in AI (Brooks, 1986). In Brooks' model, *subsumption architecture*, each module must be computationally simple and independent. These modules, now referred to as "behaviors," were originally to consist only of finite state machines. That is, there are an explicit number of states the behavior can be in, each with a characteristic, predefined output. A finite state machine also completely specifies which new states can be reached from any given state, with transitions dependent on the input to the machine.

Brooks' intent in constraining all intelligence to finite state machines was not only to simplify the engineering of the behaviors, but also to force the intelligence to be *reactive*. A fully reactive agent has several advantages. Because its behavior is linked directly to sensing, it is able to respond quickly to new circumstances or changes in the environment. This in turn allows it to be *opportunistic*. Where a conventional planner might continue to execute a plan oblivious to the fact that the plan's goal (presumably the agent's intention) had either been fulfilled or rendered impossible by other events, an opportunistic agent notices when it has an opportunity to fulfill any of its goals, and exploits that opportunity.

Two traits make the robots built under subsumption architecture highly reactive. First, each individual behavior can exploit opportunities or avoid dangers as they arise. This is a consequence of each behavior having its own sensing, and running continuously (in parallel) with every other behavior. Second, no behavior executes as a result of out-of-date information. This is because no information is stored — all information is a reflection of the current environment. Although useful for the reasons expressed, these traits also create problems for designing agents capable of complex behavior. To begin with, if there are two behaviors pursuing different goals, then it might be impossible for both to be opportunistic simultaneously. Consequently, any agent sophisticated enough to have potentially conflicting goals (such as "eat" and "escape danger") must also have some form of behavior arbitration.

Subsumption architecture provides behavior arbitration through several mechanisms. First, behaviors are organized into *layers*, each of which pursues a single goal, e.g. walking. Behaviors within the same goal are assumed not to contradict each other. Higher layers are added to lower layers with the capability to suppress their behaviors if necessary. Where behaviors within a layer might interfere with each other, another mechanism allows for pre-ordained behavior arbitration. Behaviors are allowed to suppress each other's outputs (actions) or either block or change their inputs (senses). These actions occur on communications channels between the behaviors (*wires*, originally in the literal sense), not in the behaviors themselves. All such interference is designed as part of the layer; it does not affect the inner workings of a behavior, only the expressed consequences of those workings.

After experimentation, a third mechanism of behavior selection was introduced into subsumption architecture. The description of behaviors was changed from "a finite state machine" to "a finite state machine augmented by a timer." This timer *could* be set by external behaviors, and resulted in its own behavior being deactivated until the timer ran out. The addition of the timer was required due to necessity during the development of Herbert, the can-retrieving robot (Connell, 1990). When Herbert had found a can and began to pick it up, its arm blocked its camera, making it impossible for the robot to see the can. This would allow the robot's "search" behavior to dominate its "pick up can" behavior, and the can could never be successfully retrieved.

2.4.3 An Example of Directed Architecture Evolution

Allowing the can-grasping behavior to suppress all other behaviors via a timer was a violation of reactiveness. The issue being addressed here is memory — Herbert should presumably have been able to briefly remember the sight of the can, or the decision to retrieve the can, but such a memory would violate the second trait of subsumption architecture listed above. This is a fundamental problem for fully reactive systems: they have no memory. Without memory, an agent cannot learn, or even perform advanced control or perception. Control often requires keeping track of intentions in order to exploit the result of decisions not evident in the environment. Perception often requires processing ambiguous information which can only be understood by using other recent sensor information or other stored knowledge of context to set expectations.

Brooks initially resisted having any sort of memory or learning in his architecture. The timer augmentation was the minimum possible addition to the reactive system. It registers globally (by setting the timers on each possibly interrupting behavior) that a decision has been made without making any globally accessible record of that decision. The only evidence of the chosen activity is the single unsuppressed behavior. Brooks' early resistance to the concept of learning is strikingly similar to that of the ethologist Conrad Lorenz. Both researchers initially considered intelligent behavior to be too complicated to allow for plasticity and still maintain order.

Subsequent users of subsumption architecture immediately saw the need for plasticity, and generally ignored the strict limitations on variable state. The tour-giving robot Polly (Horswill, 1993) had global variables for maintaining its current navigation goal as well as whether or not it was conducting a tour, and where that tour had begun. The musical system, the Reactive Accompanist (Bryson, 1995) had neural networks in some of its behaviors, which though theoretically finite state, were effectively variable storage of recent events. Another robot built under Brooks' direct supervision, Toto (Matarić, 1990), actually learned a map. Toto had a module that constructed a landmark-based map as individual nodes with memory. When making navigation choices, the map itself determined the best route; it was not read by an external process. Brooks (1991) acknowledges this need, and his current projects also contain neural-network type representations (Brooks et al., 1998; Marjanovic et al., 1996). This is a good example of an architectural imperative being discovered as a consequence of direct experimentation and experience.

3 Current Research Trends in AI Architectures

The architectures introduced in the previous section are still serving as models and inspiration to various members of the agent architecture community. However, they are not in themselves currently being researched extensively. In this section, we turn to several architectural paradigms for which there are sufficient numbers of practitioners to result in significant selective pressure. Some of these research communities share specific software systems which progress in published versions, others are united only by conferences and journals. To some extent, the latter approach allows for more variation, but also for less coordinated or principled change. Nevertheless, all show decisive trends, and in particular those trends outlined earlier in the introduction.

This section does not attempt a full review of the related architecture literature. Instead, we concentrate on architectures or architectural traditions that are widely-known or used. The subsections are in approximate increasing order of the size of their currently active research communities.

3.1 Behavior-Based Architectures

We begin by examining the continued evolution of behavior-based architectures. The use of the reactive and/or behavior-based approach is still widespread, particularly in academic robotics and character-based virtual reality. However, no single architecture is used by even ten percent of these researchers. Subsumption architecture, described above, is by far the best known of the architectures, but relatively few agents have been built that adhere to it strictly. For example, Matarić 1990; Bryson 1992 and Pebody 1995 all include adaptive extensions; Appleby and Steward (1994) make the behaviors nearly completely independent — they would now be called "agents". Most roboticists, even within Brooks' own laboratory, seem to have been more inspired to develop their own architecture, or to develop code without a completely specified architecture, than to attend to the details of subsumption (e.g. Horswill, 1993; Steels, 1994; Marjanovic et al., 1996; Parker, 1998).

Of the many behavior-based architectures inspired by subsumption, the one that in turn attracted the most attention has been Maes' spreading activation network (Maes, 1991a). Maes' architecture consists of a number of nodes, including action nodes, perception nodes, and goal nodes. The nodes are connected to one another by a two-way system of links. One link specifies the extent to which the second node requires the first node to have executed, the other specifies the extent to which the first node enables the second node to fire. These conduits are used to allow activation to spread both bottom-up, starting from the perception nodes, and top-down, starting from the goal nodes. When a single node gets sufficient activation (over a threshold) that node is executed.

Maes' greatest explicit hypothetical difference from subsumption architecture is her belief that agents must have multiple, manipulable goals (see Maes, 1990b). Maes' claim in that paper that subsumption architecture only allows the encoding of a single goal per agent is mistaken; however the strictly stacked goal structure of subsumption is sufficiently rigid that her arguments are still valid. A more implicit hypothesis is the need for a way to specify sequential behaviors, which her weighting of connections allows. On the other hand, Maes is very explicitly opposed to the notion of hierarchical behavior control (Maes, 1991b). Maes states that using hierarchical methods for behavior arbitration creates a bottleneck that necessarily makes such a system incapable of being sufficiently reactive to control agents in a dynamic environment.

This hypothesis was disputed by Tyrrell (1993), who showed several flaws in Maes approach, most notably that it is insufficiently directed, or in other words, does not focus attention sufficiently. There appears to be no means to set the weights between behaviors in such a way that nodes composing a particular "plan of action" or behavior sequence are very likely to chain in order. Other related behaviors often fire next, creating a situation known as "dithering". There is actually a bias against a consummatory or goal behavior being performed rather than one of its preceding nodes, even if it has been enabled, because the goal, being in a terminating position, is typically connected to fewer sources of activation.

Tyrrell's competing hypothesis is that hierarchy can be exploited in action selection, providing that all behaviors are allowed to be fully active in parallel, and that the final decision is made by combining their computation. Tyrrell refers to this strategy as a *free-flow hierarchy* and attributes it to Rosenblatt and Payton (1989). Tyrrell (1993) gives evidence for his hypothesis by comparing Maes architecture directly against several hierarchical ones, of both free-flow and and traditional hierarchies, in a purpose-built artificial life environment. In Tyrrell's test world, a small animal needs to balance a large number of often conflicting goals of very different types. For example, it must eat, maintain body temperature, sleep in its home at night, avoid two different types of predators, and mate as frequently as possible. Simulations cover up to 10 days of life and involve thousands of decision cycles per day. Using extensive experimentation, Tyrrell demonstrated substantial advantage to all the hierarchical architectures over Maes approach.

Tyrrell also shows statistically significant superiority of the free-flow hierarchy over its nearest strictly-hierarchical competitor, which was in fact the most simple one, a drive-based model of control. He claims that free-flow hierarchy must be an optimal action selection mechanism, because it is able to take into account the needs of all behaviors. These sorts of cooperative rules have been further refined. For example, Humphrys (1997) suggests choosing a course that minimizes the maximum unhappiness or disapproval of the elements tends to lead to the optimal solutions. Such thoroughly distributed approaches have been challenged by Bryson (1999b). Bryson suggests that simplicity in finding an optimal design, whether by a programmer or by a learning process such as evolution, outweighs the advantage of cooperative negotiation. Bryson's behavior system uses a hierarchical controller where only a small subset of nodes, corresponding in number to the elements in the top layer in the hierarchy, actively vie for control of the agent. Further, these nodes do not compete on the basis of relative activation levels, but are activated by threshold and strictly prioritized. Thus on any particular cycle, the highest priority node that has threshold activation takes control. Within the winner's branch of the hierarchy, further competitions then take place. This is very similar to a traditional hierarchy, excepting parallel roots and some other details of execution, yet Bryson (1999b) shows a statistically significant improvement over the Tyrrell (1993) results using the same system for evaluation.

Blumberg (1996) presents another architecture which takes considerable inspiration from both Maes and Tyrrell, but also extends the control trend further towards conventional hierarchy. Blumberg's system, like Tyrrell's, organizes behaviors into a hierarchy while allowing them to be activated in parallel. However, in Blumberg's system the highest activated module wins and locks any critical resources it requires, such as legs if the module regulates walking. Nodes that are also active but do not require locked resources are allowed to express themselves. Thus a dog can both walk and wag its tail at the same time for two different reasons. The hierarchy is also exploited to focus attention in the voting system. Not every behavior participates in the vote: a fact that was initially minimalized (Blumberg, 1996), but more recently has become a stated feature of the system (Kline and Blumberg, 1999). Blumberg's architecture is being used by his own and other research groups (including Brooks' (Breazeal and Scassellati, 1999)) as well as a major commercial animation corporation, so its future development should be of significant interest.

Summary

All behavior-based systems are modular; the modular design strategy to a large part defines the paradigm. Most behavior-based systems rely on their modularity as their source of reactiveness — any particular behavior may express itself opportunistically or when needed. This has however lead to difficulties in action selection which seem to have limited the complexity of the tasks addressed by these systems. Action selection mechanisms vary widely between individual architectures, indicating the field has not settled on a stable solution. However, several architectures are now incorporating hierarchical and or sequential elements.

3.2 Multi-Layered Architectures

The achievements of behavior-based and reactive AI researchers have been very influential outside of their own communities. In fact, there is an almost universal acceptance that at least some amount of intelligence is best modeled in these terms, though relatively few would agree that all cognition can be described this way. Many researchers have attempted to establish a hybrid strategy, where a behavior-based system is designed to work with a traditional AI planner, which deduces the next action by searching a knowledge base for an act that will bring it closer to a goal. Traditionally, planners have micro-managed, scripting every individual motion. By making their elements semi-autonomous behaviors which will react or adapt to limited uncertainty, the planners themselves can be simplified. The following is a recent account of a project from the late 1980s:

> "The behavior-based plan execution was implemented bottom up to have as much useful capability as possible, where a useful capability is one which looked like it would simplify the design of the planner. Similarly, the planner was designed top down towards this interface, clarifying the nature of useful capabilities at which the behavior-based system should aim. This design method greatly reduced the complexity of the planner, increasing the complexity of the agent much less than this reduction, and thus reduced the overall system complexity. It also produced a robust system, capable of executing novel plans reliably despite... uncertainty." (Malcolm, 1997, Section 3.1)

Malcolm's system can be seen as a two-layer system: a behavior-based foundation controlled by a planning system. More popular of late have been three-layer systems. These systems are similar, except that there is a middle layer that consists of precoded plan fragments, sometimes referred to as "implicit knowledge", in contrast to the "explicit" reasoning by the top-level planner. Another distinction is that the middle layer is often considered reactive, in that it does not create plans, but selects them based on the situation; while the top layer is a traditional constructive planner. In most systems, the top-layer planner manipulates or generates this intermediate representation level rather than acting directly on the behavior primitives.

The currently dominant layered robot architecture is probably 3T (Bonasso et al., 1997), which features Reactive Action Packages (RAP, Firby, 1987), for its middle layer. RAP is an architecture for creating reactive, flexible, situation-driven plans, and itself uses a lower layer of behavior primitives. 3T integrates this system with a constructive planner. 3T has been used on numerous robots, from academic mobile robots, to robotic arms used for manipulating hazardous substances, previously controlled by teleoperation, to maintenance robots for NASA's planned space station. Leon et al. (1997) uses 3T in simulation to run an entire space station, including farming and environmental maintenance. See Hexmoor et al. (1997) and Kortenkamp et al. (1998) for recent reviews of many two- and three-layer architectures.

3T may seem a more likely tool for modeling of human-like intelligence than the behavior-based models discussed earlier, in that it has something approximating logical competence. However, planning has been mathematically proven an unrealistic model of intelligence because it relies on search (Chapman, 1987). Search is combinatorially explosive: more behaviors or a more complex task leads to an exponentially more difficult search. Though there is no doubt that animals do search in certain contexts (e.g. seeking food, or for a human, choosing a gift), the search space must be tightly confined for the strategy to be successful. A better model of this sort of process is ATLANTIS (Gat, 1991), which is controlled by its middle layer, and only operates its top, planning layer on demand. This model is in fact quite similar to the Norman and Shallice (1986) model of human action selection, where conscious control is essentially interrupt driven, triggered by particularly difficult or dangerous situations. Although the alternative model, with the top level being the main controller, is more typical (Bonasso et al., 1997; Albus, 1997; Hexmoor et al., 1997; Malcolm, 1997), Gat's model would also seem a more natural extension of the behavior-based approach. It is also notable that Bonasso et al. (1997) report a number of 3T projects completed using only the lower two layers.

Another incompatibility between at least early behavior-based work and the layered system approach is the behavior-based systems' emphasis on emergence. For a hybrid system, emergent behavior is useless (Malcolm, 1997). This is because an emergent behavior definitionally has no name or "handle" within the system, consequently the planning layer cannot use it. In humans at least, acquired skills can be recognized and deliberately redeployed (Karmiloff-Smith, 1992). Hexmoor (1995) attempts to model both the development of a skill (an element of the middle layer) from actions performed deliberately (planned by the top layer) and the acquisition of deliberate control of skills. His hypothesis of requiring both these forms of learning are probably valid, but his actual representations and mechanisms are still relatively unproven. Another group researching the issue of learning behaviors and assigning their levels is that of Stone and Veloso (1999). Veloso's group has had a series of highly successful entrants into various leagues of robot soccer; their architecture is thus also under strenuous selective pressure. It also seems to be converging to modularity in the areas which are most specialized, such as communication and learning, while having a DAG for action selection over preset plans.

Summary

Two- and three-layer architectures succeed at complex tasks in real environments. They generally have plans and plan hierarchies at their second layer, carefully organized into reactive plans in order to maintain reactivity, although some architectures rely on the bottom-level behaviors for this function, and others do not operate in dynamic environments. Modularity has generally been limited to the lower level, though in some architectures the top-level planner can also be seen as a specialized module. Current research indicates there are still open questions concerning the optimal kind of planning for the top layer, and how to manipulate and shift information between representations, particularly learned skills.

3.3 PRS — Beliefs, Desires and Intentions

Although robotics has been dominated by three-layer architectures of late, the field of autonomous agents is dominated, if by any single architecture, by the Procedural Reasoning System, or PRS (Georgeff and Lansky, 1987; d'Inverno et al., 1997). PRS also began as a robot architecture, but has proven sufficiently reliable to be used extensively for tasks such as aircraft maintenance and defense simulations. It was originally developed at roughly the same time as subsumption architecture, as a part of a follow-up program to the longest running robot experiment ever, Shakey (Nilsson, 1984). PRS is designed to fix problems with traditional planning architectures exposed by the Shakey project. Such problems include:

- Forming a complete plan before beginning action. This is a necessary part of the search process underlying planning — a planner cannot determine whether a plan is viable before it is complete. Many plans are in fact formed backwards: first selecting the last action needed to reach the goal, then the second last and so on. However, besides the issues of opportunism already discussed, many details of a real problem cannot be known until the plan is executed. For example, when crossing a room full of people, the locations of the people are determined at the time of crossing, and cannot be predetermined.
- Taking too long to create a plan, ignoring the demands of the moment. The standard example is trying to cross a road — a robot will not have time to replan if it suddenly spots a car, it will need to know reactively to move out of the way.

- Being unable to create plans that contain elements other than primitive acts — to take advantage of skills or learned procedures.
- Being unable to manipulate plans and goals. Plans may need to be abandoned, or multiple goals pursued simultaneously.

Obviously, this list is very similar to the problems the behavior-based programmers attempted to solve. There are two main differences. First, PRS maintains as a priority the ability to construct plans of action. The architecture allows for incorporating specialized planners or problem solvers. The second difference is that PRS development is couched very much in psychological terms, the opposite of Brooks' deprecation of conscious impact on intelligent processes. PRS is referred to as a BDI architecture, because it is built around the concepts of beliefs, desires and intentions.

Many researchers appreciate the belief, desires and intentions approach in concept, without embracing PRS itself. For example, Sloman and Logan (1998) consider the notions of belief, desire, intention and emotion as central to an agent, but proposes expressing them in a three-layer architecture. Sloman's top layer is reflective, the middle deliberative, and the bottom layer reactive. This is similar to Malcolm (1997) or the first and third layer of 3T (Bonasso et al., 1997), but with an additional layer dedicated to manipulating the goals of their top layers, and considering its own current effectiveness. This particular role assignment for the layers of a three-layer architecture is also proposed in Figure 2, below.

The PRS architecture consists of four main components connected by an interpreter (sometimes called the "reasoner") which drives the processes of sensing, acting, and rationality. The first component is a database of *beliefs*. This is knowledge of the outside world from sensors, of the agent's own internal states, and possibly knowledge introduced by outside operators. It also includes memories built from previous knowledge. The second component is a set of *desires*, or goals. These take the form of behaviors the system might execute, rather than descriptions of external world state as are often found in traditional planners. The third PRS component is a set of *plans*, also known as knowledge areas. Each plan is not necessarily completely specified, but is more likely to be a list of subgoals useful towards achieving a particular end, somewhat like the hierarchical reactive plans described earlier in the context of Bryson (1999b). These may include means by which to manipulate the database (beliefs) to construct a next action or some new knowledge. The final main component is a stack of *intentions*. Intentions are simply the set of plans currently operating. A stack indicates that only one plan is actually driving the command system at a time, but multiple plans may be on the stack. Typically, ordering of the stack is only changed if one plan is interrupted, but new information may trigger a reorganization.

Like multi-layer architectures, PRS works from the hypothesis that a system needs both the ability to plan in some situations, such as navigation, and the ability to execute skilled acts for situations where search is not reasonable, such as avoiding trucks. In some sense, each plan is like a behavior in behavior-based AI. Behavior-based AI is essentially a retreat to allowing programmers to solve the hard and important problems an agent is going to face in advance. A procedure to solve an individual problem is usually relatively easy to design. Thus some modularity can be found in the design of the knowledge areas that make up the plan library. On the other hand, PRS does not see specialized state and representations dedicated to particular processes as worth the tradeoff from having access to general information. It has moved the procedural element of traditional planners closer to a behavior-based ideal, but only allows for specialized or modularized data by tagging. The interpreter, goal list and intention stack are the action selection device of PRS.

PRS exists as a planning engine and a set of development tools. It exists in several versions and can be acquired from a number of different sites. It is used by industry and the US government as well as for research. One large, relatively recent change in the basic structure has been the adoption of a formalism for its plan libraries that can also be used by a conventional constructive planner (Wilkins et al., 1995). This move can be seen as a part of a general trend in current PRS research to attempt to make the system easier to use — the idea of a planner is to allow plan libraries to be generated automatically. The majority of this research is concerned with easing the design task under PRS. There is now a "PRS-lite" (Myers, 1996), and a number of tool benches and formal methods for proving code correct (e.g Huber, 1999; d'Inverno et al., 1997). The original development lab for PRS, SRI, is now focusing effort on a much more modularized AI architecture, built under a multi-agent paradigm (Wilkins and Myers, 1998).

The pre-history of PRS, the Shakey project, also has relevant evolutionary trends (Nilsson, 1984). Although Shakey had a traditional planner (called STRIPS), over the term of the project the concept of *triangle tables* was developed. A triangle table decomposes a plan into its steps and assumptions, then creates a contingency table allowing the plan to be restarted from any point. Perception is then used to determine which element of the plan should be executed next. This allows action selection to be reactive within the confines of the plan, rather than relying on memory of what steps should have already been executed. This approach leads naturally into teleo-reactive plans (Nilsson, 1994), another recently developed form of storage for skilled behaviors developed by planners. Benson (1996) describes using this as the basis of a system that learns to fly airplanes in flight simulators, and the architecture is being used at a number of research laboratories.

The Shakey project also moved from having multiple world models in its first implementation to having a single storage place for predicates of observed data. Any predicate used to form a new plan was rechecked by observation. This development under the selective pressure of experimentation lends credence to the idea that too much modeling of the world is a likely cause of difficulties.

Summary

PRS and its related BDI architectures have been much more popular than behavior-based systems, particularly outside of academic settings. This may be because they are easier to program. They provide significant support for developing the action-selection mechanism, a hierarchical library of plans, and a separate, specialized mechanism for reprioritizing the agents attention in response to the environment. Particularly when taken over their long-term history, however, these architectures have converged on some of the same important principles such as simplified representations (though not specialized ones) and modularization (at least in the plan libraries.) Current research trends indicate that designing the agent is still a critical problem.

3.4 Soar and ACT-R

Soar (Newell, 1990) and ACT-R (Anderson, 1993) are the AI architectures currently used by the largest number of researchers, not only in AI, but also in psychology and particularly cognitive science. These architectures are fundamentally different from the previously reviewed architectures. Both are also older, dating to the late 1970s and early 1980s for their original versions, but both are still in active development (Laird and Rosenbloom, 1994; Anderson and Matessa, 1998). The Soar community in particular has responded to the behavior-based revolution, both by participating directly in competitions with the approach (Kitano et al., 1997) and even by portraying their architecture in three layers (see Figure 2).

— Figure 2 about here. —

Soar and ACT-R both characterize all knowledge as coming in two types: data or procedures. Both characterize data in traditional computer science ways as labeled fields, and procedures in the form of production rules.

Soar is a system that learns to solve problems. The normal procedure is to match its production rules against the current state of the world, find one that is applicable, and apply it. This is automatic, roughly equivalent to the middle or bottom layer of a three-layer architecture. If more than one production might work, or no production will fire, or nothing has changed since the previous application of a production, then Soar considers itself to be at an *impasse*. When Soar encounters an impasse, it enters a new problem space of trying to solve the impasse rather than the current goal. The new problem space may use any means available to it to solve the problem, including planning-like searches. Soar has several built-in general purpose problem solving approaches, and uses the most powerful approach possible given the current amount of information. This process is thus something like the top level of

ATLANTIS (Gat, 1991), except that Soar allows the process to recurse, so the meta-reasoner can itself hit an impasse and another new reasoning process is begun.

Soar includes built-in learning, but only of one type of information. When an impasse is resolved, the original situation is taken as a precondition and the solution as a procedure, and a new rule is created that takes priority over any other possible solution if the situation is met again. This is something like creating automatic skills out of declarative procedures, except that it happens quickly, on only one exemplar. This learning system can be cumbersome, as it can add new rules at a very high rate, and the speed of the system is inversely related to the number of rules. To partially address this problem, Soar has the concept of a *problem space*, a discrete set of productions involved in solving a particular goal or working in a particular context. This makes the system roughly hierarchical even in its non-impasse-solving mode.

ACT-R is essentially simpler than Soar: it does not have the impasse mechanism nor does it learn new skills in the same way. Nevertheless, ACT-R is used extensively for cognitive modeling, and has been used to replicate many psychological studies in decision making and categorization (Anderson, 1993). ACT-R also faces the difficulty of combinatorics, but it takes a significantly different approach: it attempts to mimic human memory by modeling the probability that a particular rule or data is recalled. Besides the two sets of "symbolic" knowledge it shares with Soar, ACT-R keeps Bayesian statistical records of the contexts in which information is found, its frequency, recency and utility (Anderson and Matessa, 1998). It uses this information to weight which productions are likely to fire. It also has a noise factor included in this statistical, "sub-symbolic" system, which can result in less-likely alternatives being chosen occasionally, giving a better replication of the unpredictability of human behavior. Using alternatives is useful for exploring and learning new strategies, though it will often result in suboptimal performance as most experiments prove to be less useful than the best currently-known strategy.

Soar, like PRS, is used on an industrial level. However, the fact that it is losing popularity within the cognitive science research community to ACT-R is attributed by researchers largely to the the fact that ACT-R is significantly easier to work with. This is largely because Soar was designed primarily to learn — researchers compared programming Soar to teaching by brain surgery. One simplification in ACT-R had to be done away with, however. Originally it did not have problem spaces, but over the course of research it was found that hierarchical focusing of attention was necessary to doing anything nearly as complex as modeling human mathematical competences, the primary goal of ACT-R's development team (Anderson, 1993).

Soar has also evolved significantly (Laird and Rosenbloom, 1994). In particular, when moving to solve problems in a dynamic, real-world domain, it was found to be critical to allow programmers to specify chains or sequences of events explicitly, rather than in terms of simple productions. The encoding of time and duration was another major challenge that had to be overcome when Soar moved into robotics

— a problem that also needed to be addressed in early versions of PRS and RAP, the middle layer of 3T (Myers, 1996). ACT-R has not yet been adapted to the problems of operating in a dynamic world: representing noisy and contradictory data, and reasoning about events over time.

Summary

Despite coming from significantly different perspectives and research communities, the long and well-documented histories of Soar and ACT-R exhibit many of the same trends as the other paradigms previously examined. Since both systems are very distributed, (their control is based almost entirely on production rules) they are necessarily very reactive. In fact, Soar had to compromise this feature to be able to provide real-time control. Modularity of control if not data is provided in problem spaces, which can be hierarchical, and Soar now provides for explicit sequential action selection. Soar's generic representations were also found to be not entirely satisfactory. There has been forced specialization of procedure types due to the new benchmark tasks of the 1990's, particularly mobile robotics. Soar still suffers from an extreme overhead in programming difficulty, but is also still in widespread use. ACT-R exploits a niche as a simpler though similar form of learning system, and has been further specialized to improve its ability to model human cognition.

4 Discussion and Recommendations

There have been complaints within the autonomous control community about the over-generation of architectures: what is wanted by users are improvements on systems with which they are already familiar, rather than a continuous diversification. This argument contains some truth, however it overlooks the perspective stated in the introduction. An agent architecture is a design methodology, and a design methodology is not simply a piece of software. Although some architectural features will conflict, in many cases there is no reason architectures cannot be combined, or one architecture implemented within another. We have already demonstrated this principle by two combining relatively minor behavior-based architectures (Bryson, 1999a; Thórisson and Bryson, tion) and are currently working on establishing idioms within PRS to better exploit reactive planning. Some of the insights from this work are discussed below.

Behavior-based architectures began with many of the advantages of modularity and reactive systems, but development of complex control software in them has been hampered by the lack of specific control architectures for supporting hierarchical and sequential ordering of action selection. This is largely due to theoretical opposition: Is a system truly autonomous if it is forced to carry out a plan? Is centralized control biologically plausible? The answer to both of these questions is almost certainly "yes", but space does not permit the arguments here — see for example Barber and Martin (1999) and Bryson

(2000) respectively for some discussion. Regardless, it can be observed empirically that all autonomous agents do still have and require action selection mechanisms. In behavior-based systems, these systems are often distributed across the behaviors. This may lead to some improvement of robustness, but at a considerable cost in programmability and ease of debugging.

The shift to layered architectures may therefore seem a natural progression for behavior-based AI, but we have some reservations about this model. As mentioned above, many of the systems have the deliberate or constructive planner in ultimate control, which may be intuitive but has not yet been demonstrated correct. The frequent lack of such a layer within this research tradition, and the success of PRS and Soar with something more like a middle layer in primary control of action selection are good indications that primary action selection should probably emphasize reactive planning rather than deliberation.

A further concern is that layered systems, and indeed some of the more recent behavior-based systems such as HAP (Bates et al., 1992; Reilly, 1996) or the free-flow hierarchy architectures reviewed above, have denigrated the concept of a "behavior" to a mere programming language primitive, thus losing much of the advantage of modularity. Blumberg (1996) addressed this by creating "clusters of behaviors"; We would argue that these clusters are at the more appropriate level for a behavior.

Behaviors were originally designed as essentially autonomous entities that closely coupled perception and action to achieve a particular competence. Unfortunately, they were also conceived as finite state machines, with no internal variable state. In nature, perception is universally accompanied by memory and learning: much of development in mammals is dedicated to learning to categorize and discriminate. We would argue that behaviors should therefore also contain state appropriate to their competence, and further that this state and learning should be at the center of behavior decomposition, much as it is at the center of modern object decomposition in object-oriented design. We are currently developing this concept of "behavior-oriented design" for AI; some early work on these ideas is already available (Bryson, 1995; Bryson and McGonigle, 1998).

In general, we suspect that the two- or three-layered architectures are still an over-simplification of the number of modules needed to have a complete architecture. Besides constructive and reactive planners, other systems that have proven useful in autonomous agents are an emotional or affective system (e.g. Reilly, 1996; Sloman and Logan, 1998), or an autonomous scheduler which selects the appropriate expression of a selected action (Thórisson, 1999). Our primary suggestion for behavior-based AI is further attention to easing the design of action selection. We also suggest experimenting with limited functional modules for abilities such as operating sequential plans and smoothing motor output. This development would be parallel to the nearly universal, though still reductionist, use of state in this paradigm.

Our suggestion for three-layered architectures is that they look for ways to increase support of modularity in their systems. We are also concerned about the reactiveness of systems not using a middle layer specifically designed to compensate for dynamic environments, such as RAP. It is still not clear whether it is a better idea for a system to separate action selection from reactivity concerns, as PRS does, rather than using one system for both, as do 3T and the behavior-based architectures which use hierarchical action selection (e.g. Bryson and McGonigle, 1998).

PRS is in some ways similar to a three-layer architecture with the emphasis on the middle layer — the building of the plan library. In particular, the software version of PRS distributed by SRI has a fairly impressive GUI for supporting the editing and debugging of this level of intelligence. As might be gathered from our discussion of three-layer architectures above, we consider this type of support very useful.

Unfortunately, PRS still leaves two important levels of abstraction largely unsupported, and somewhat difficult to manage. The construction of primitives is left to the user, to be done in the language of the PRS implementation; in the case of SRI's implementation, this is common lisp. Our suggestion for behavior-based and three-layer architectures applies equally here: primitives should be ordered modularly. They can in fact be built from methods on objects with proprietary state, not shared by the PRS database system. We recognize that this might offend PRS purists, particularly because it might have consequences for the theoretical work on proving program correctness that relies on the database. Nevertheless, we stand by our claim that state is a part of perception. Having some state proprietary to a module should be no more difficult than having an external sensor proprietary to a primitive function; in fact it is exactly equivalent.

The other design level that is surprisingly neglected is the hierarchical organization and prioritization of the various elements of the plan library. Although it is possible to organize plans in the file space (a collection of plans may be saved in a single file) and in lisp by placing them in packages, there is no GUI tool that allows for viewing more than one plan at a time. There is no tool for ordering plans within clusters or agents. Consequently, there is no visual idiom for prioritizing plans that might otherwise be simultaneously able to fire. Prioritization must be handled in lisp code that is triggered during the meta-rule section of the main processing cycle. Providing a tool to address this would make it far simpler to program a reactive plan structure (e.g. Shakey's triangle tables (Nilsson, 1984), Nilsson's teleo-reactive plans (Nilsson, 1994), or Bryson's competences (Bryson and McGonigle, 1998)), which seems fundamental to true reactive planning.

It should be noted that PRS-lite actually addresses both of these complaints, though not in the manner recommended above. It supports "fuzzy" behaviors as primitives, which have their own design methodology, and it attempts to eliminate the need for meta-reasoning or prioritization by a combination of simplifying the task and increasing the power of the goal descriptions (Myers, 1996). Whether these solutions prove adequate, the fact that these areas are a focus of change indicates agreement on the areas of difficulty in using PRS.

Of the paradigms reviewed, we have the least personal experience with Soar and ACT-R, having only experienced them through tutorials and the anecdotes of programmers. Given their very different background and structure, they appear to have remarkably similar design issues to those experienced under the early behavior-based architectures. This is perhaps unsurprising since both systems are thoroughly distributed. The parallel between the story of the augmenting of subsumption architecture recounted above and the story of the augmentation of Soar with time and sequencing in order to facilitate robot control recounted in Laird and Rosenbloom (1994) is also striking. Our suggestions for improving Soar are consequently essentially our recommendations for agent architectures in general: to focus on making agents easier to design via enhancing the ease of use of modular decomposition and pre-programmed action selection, while still maintaining Soar's provision for reactivity and opportunism.

Conclusions

Every autonomous agent architecture seems to need:

- A modular structure and approach for developing the agent's basic behaviors, including perception, action and learning.
- A means to easily engineer individual competences for complex tasks. This evidently requires a means to order action selection in both sequential and hierarchical terms, using both situation-based triggers and agent-based priorities derived from the task structure.
- A mechanism for reacting quickly to changes in the environment. This generally takes the form of a system operating in parallel to the action selection, which monitors the environment for salient features or events.

In addition to the technical requirements just listed, the central theme of this article is that agent architectures are first and foremost design methodologies. The advantages of one strategy over another are largely a consequence of how effectively programmers working within the approach can specify and develop the behavior of the agent they are attempting to build. It should be noted that this stance is not necessarily antithetical to concerns such as biological plausibility or machine learning: natural evolution and automatic learning mechanisms both face the same problems of managing complexity as human designers. The sorts of bias that help a designer may also help these other processes. Similarly, where it is understood, natural intelligence serves as a knowledge source just as well as any other successful agent. Our intention with this article is to draw attention to the knowledge developed in our field over the past two decades. In particular, we have attempted to cross boundaries between significantly different paradigms of agent architecture development, and present useful recommendations for each of these various approaches.

Acknowledgments

As stated in the Introduction, much knowledge about agent architectures is not necessarily obvious from the literature, but can only be found through experience. Acknowledgment is due to everyone who has held long, contentful conversations with me on these topics. Particular thanks to Kris Thórisson, Paolo Petta, Push Singh, Ian Horswill, Marcel Schoeppers, Michael Mateas, John Laird, and Leslie Pack Kaelbling. Thanks to Chris Wright, Jered Floyd, Olin Shivers, Lynn Stein, Henry Hexmoor, Brendan McGonigle, John Hallam and the anonymous reviewers for their proofreading and suggestions on this paper. The agent in Figure 1 was drawn by Will Lowe.

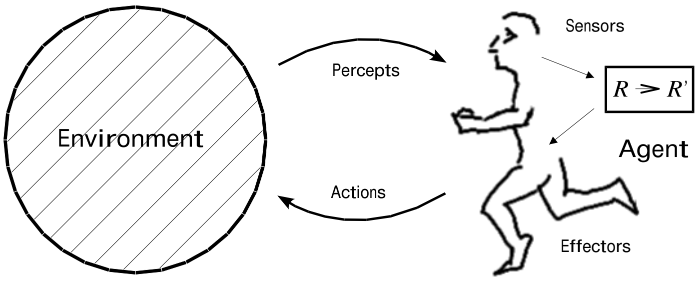

Figure 1: A traditional AI architecture (after Russell and Norvig, 1995).

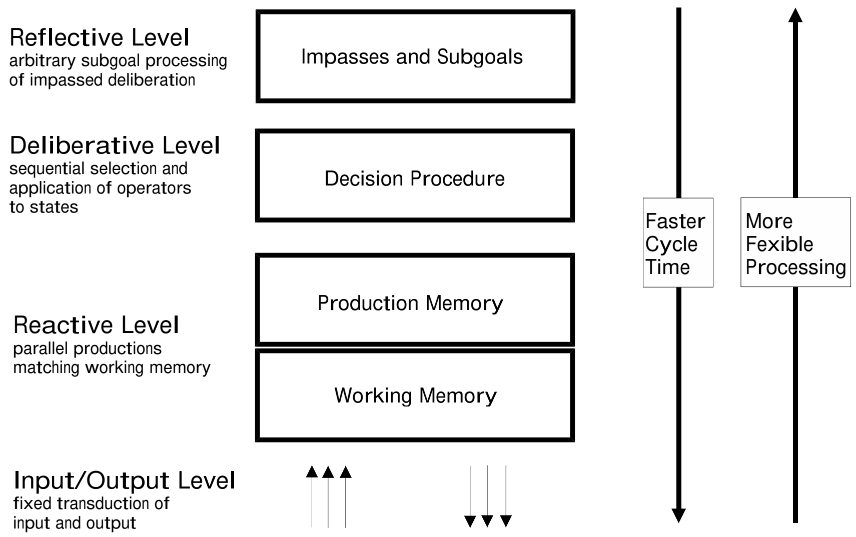

Figure 2: Soar as a three-layer architecture) (after Laird and Rosenbloom, 1994).

References

Adams, J. A. (1984). Learning of movement sequences. *Psychological Bulletin*, 96(1):3–28.

Albus, J. S. (1997). The NIST real-time control system (RCS): an approach to intelligent systems research. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2/3):147–156.

Anderson, J. R. (1993). *Rules of the Mind*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Anderson, J. R. and Matessa, M. (1998). The rational analysis of categorization and the ACT-R architecture. In Oaksford, M. and Chater, N., editors, *Rational Models of Cognition*. Oxford University Press.

Appleby, S. and Steward, S. (1994). Mobile software agents for control in telecommnications networks. *BT Technology Journal*.

Baerends, G. P. (1976). The functional organisation of behaviour. *Animal Behaviour*, 24:726–738.

Barber, K. S. and Martin, C. E. (1999). Agent autonomy: Specification, measurement, and dynamic adjustment. In *Proceedings of the Autonomy Control Software Workshop at Autonomous Agents (Agents'99)*, pages 8–15, Seattle, WA.

Bates, J., Loyall, A. B., and Reilly, W. S. (1992). An architecture for action, emotion, and social behavior. Technical Report CMU-CS-92-144, CMU School of Computer Science, Pitsburgh, PA.

Beer, R. D. (1995). Computational and dynamical languages for autonomous agents. In Port, R. F. and van Gelder, T., editors, *Mind as Motion: Explorations in the Dynamics of Cognition*, chapter 5, pages 121–147. MIT Press, Cambridge, MA.

Benson, S. (1996). *Learning Action Models for Reactive Autonomous Agents*. PhD thesis, Stanford University. Department of Computer Science.

Bizzi, E., Giszter, S. ., Loeb, E., Mussa-Ivaldi, F. A., and Saltiel, P. (1995). Modular organization of motor behavior in the frog's spinal cord. *Trends in Neuroscience*, 18:442–446.

Blumberg, B. M. (1996). *Old Tricks, New Dogs: Ethology and Interactive Creatures*. PhD thesis, MIT. Media Laboratory, Learning and Common Sense Section.

Bohm, D. (1980). *Wholeness and the Implicate Order*. Cox & Wyman.

Bonasso, R. P., Firby, R. J., Gat, E., Kortenkamp, D., Miller, D. P., and Slack, M. G. (1997). Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2/3):237–256.

Breazeal, C. and Scassellati, B. (1999). How to build robots that make friends and influence people. In *IROS-99*.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2:14–23.

Brooks, R. A. (1991). Intelligence without reason. In *Proceedings of the 1991 International Joint Conference on Artificial Intelligence*, pages 569–595.

Brooks, R. A., Breazeal, C., Irie, R., Kemp, C. C., Marjanovic, M., Scassellati, B., and Williamson, M. (1998). Alternate essences of intelligence. In *AAAI98*.

Bruner, J. S. (1982). The organisation of action and the nature of adult-infant transaction. In von Cranach, M. and Harré, R., editors, *The Analysis of Action*, pages 313–328. Cambridge University Press.

Bryson, J. (1992). The subsumption strategy development of a music modelling system. Master's thesis, University of Edinburgh. Department of Artificial Intelligence.

Bryson, J. (1995). The reactive accompanist: Adaptation and behavior decomposition in a music system. In Steels, L., editor, *The Biology and Technology of Intelligent Autonomous Agents*. Springer-Verlag.

Bryson, J. (1999a). Creativity by design: A behaviour-based approach to creating creative play. In Nack, F., editor, *AISB'99 Symposium on Creativity in Entertainment and Visual Art*, pages 9–16, Sussex. The Society for the Study of Artificial Intelligence and the Simulation of Behaviour.

Bryson, J. (1999b). Hierarchy and sequence vs. full parallelism in action selection. In Ballin, D., editor, *Intelligent Virtual Agents 2*.

Bryson, J. (2000). The study of sequential and hierarchical organisation of behaviour via artificial mechanisms of action selection. M.Phil. thesis.

Bryson, J. and McGonigle, B. (1998). Agent architecture as object oriented design. In Singh, M. P., Rao, A. S., and Wooldridge, M. J., editors, *The Fourth International Workshop on Agent Theories, Architectures, and Languages (ATAL97)*. Springer-Verlag.

Byrne, R. W. and Russon, A. E. (1998). Learning by imitation: a hierarchical approach. *Brain and Behavioral Sciences*, 21(5):667–721.

Carlson, N. R. (1994). *Physiology of Behavior*. Allyn and Bacon, Boston, 5 edition.

Chapman, D. (1987). Planning for conjunctive goals. *Artificial Intelligence*, 32:333–378.

Chomsky, N. (1957). *Syntactic Structures*. Mouton.

Connell, J. (1990). *Minimalist Mobile Robotics: A Colony-style Architecture for a Mobile Robot*. Academic Press, Cambridge, Massachusetts. also MIT TR-1151.

Dawkins, R. (1976). Hierarchical organisation: A candidate principle for ethology.". In Bateson, P. P. G. and Hinde, R. A., editors, *Growing Points in Ethology*, pages 7–54. Cambridge University Press, Cambridge.

d'Inverno, M., Kinny, D., Luck, M., and Wooldridge, M. (1997). A formal specification of dMARS. In Singh, M. P., Rao, A. S., and Wooldridge, M. J., editors, *Proceedings of the 4th International Workshop on Agent Theories, Architectures and Languages*, Providence, RI. Spring-Verlog.

Firby, J. (1987). An investigation into reactive planning in complex domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 202–207.

Fodor, J. A. (1983). *The Modularity of Mind*. Bradford Books. MIT Press, Cambridge, MA.

Freud, S. (1900). *The Interpretation of Dreams*. Norton.

Gardner, M. and Heyes, C. (1998). Splitting lumping and priming. *Brain and Behavioral Sciences*, 21(5):695.

Gat, E. (1991). *Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots*. PhD thesis, Virginia Polytechnic Institute and State University.

Georgeff, M. P. and Lansky, A. L. (1987). Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 677–682, Seattle, WA.

Goldfield, E. C. (1995). *Emergent Forms: Origins and early development of human action and perception*. Oxford University Press.

Greenfield, P. M. (1991). Language, tools and brain: The ontogeny and phylogeny of hierarchically organized sequential behavior. *Brain and Behavioral Sciences*, 14:531–595.

Hendriks-Jansen, H. (1996). *Catching Ourselves in the Act: Situated Activity, Interactive Emergence, Evolution, and Human Thought*. MIT Press, Cambridge, MA.

Henson, R. N. A. (1996). *Short-term Memory for Serial Order*. PhD thesis, University of Cambridge. St. John's College.

Hexmoor, H., Horswill, I., and Kortenkamp, D. (1997). Special issue: Software architectures for hardware agents. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2/3).

Hexmoor, H. H. (1995). *Representing and Learning Routine Activities*. PhD thesis, State University of New York at Buffalo.

Horswill, I. D. (1993). *Specialization of Perceptual Processes*. PhD thesis, MIT, Department of EECS, Cambridge, MA.

Houghton, G. and Hartley, T. (1995). Parallel models of serial behavior: Lashley revisited. *PSYCHE*, 2(25).

Huber, M. J. (1999). JAM: A BDI-theoretic mobile agent architecture. In *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 236–243, Seattle.

Hull, C. (1943). *Principles of Behaviour: an Introduction to Behaviour Theory*. D. Appleton-Century Company.

Hume, D. (1748). *Essay Concerning Human Understanding*. London.

Humphrys, M. (1997). *Action Selection methods using Reinforcement Learning*. PhD thesis, University of Cambridge.

Jorion, P. J. M. (1998). A methodological behaviourist model for imitation. *Brain and Behavioral Sciences*, 21(5):695.

Karmiloff-Smith, A. (1992). *Beyond Modularity: A Developmental Perspective on Cognitive Change*. MIT Press, Cambridge, MA.

Kelso, J. S. (1995). *Dynamic Patterns: The Self-Organization of Brain and Behavior*. MIT Press.

Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., and Osawa, E. (1997). RoboCup: The robot world cup initiative. In *Proceedings of The First International Conference on Autonomous Agent*. The ACM Press.

Kline, C. and Blumberg, B. (1999). The art and science of synthetic character design. In *AISB'99 Symposium on AI and Creativity in Entertainment and Visual Art*.

Kortenkamp, D., Bonasso, R. P., and Murphy, R., editors (1998). *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*. MIT Press, Cambridge, MA.

Laird, J. E. and Rosenbloom, P. S. (1994). The evolution of the Soar cognitive architecture. Technical Report CSE-TR-219-94, Department of EE & CS, University of Michigan, Ann Arbor. also in *Mind Matters*, Steier and Mitchell, eds.

Lashley, K. S. (1951). The problem of serial order in behavior. In Jeffress, L. A., editor, *Cerebral mechanisms in behavior*. John Wiley & Sons.

Leon, V., Kortenkamp, D., and Schreckenghost, D. (1997). A planning, scheduling and control architecture for advanced life support systems. In *NASA Workshop on Planning and Scheduling for Space*.

Mac Aogáin, E. (1998). Imitation without attitudes. *Brain and Behavioral Sciences*, 21(5):696–697.

Maes, P., editor (1990a). *Designing Autonomous Agents : Theory and Practice from Biology to Engineering and back*. MIT Press, Cambridge, MA.

Maes, P. (1990b). Situated agents can have goals. In Maes, P., editor, *Designing Autonomous Agents : Theory and Practice from Biology to Engineering and back*, pages 49–70. MIT Press, Cambridge, MA.

Maes, P. (1991a). The agent network architecture (ana). *SIGART Bulletin*, 2(4):115–120.

Maes, P. (1991b). A bottom-up mechanism for behavior selection in an artificial creature. In Meyer, J.-A. and Wilson, S., editors, *From Animals to Animats*, pages 478–485, Cambridge, MA. MIT Press.

Malcolm, C. (1997). A hybrid behavioural/knowledge-based approach to robotic assembly. DAI Research Paper 875, University of Edinburgh, Edinburgh, Scotland.

Marjanovic, M., Scassellati, B., and Williamson, M. (1996). Self-taught visually-guided pointing for a humanoid robot. In Maes, P., Matarić, M. J., Meyer, J.-A., Pollack, J., and Wilson, S. W., editors, *From Animals to Animats 4 (SAB96)*, Cambridge, MA. MIT Press.

Matarić, M. J. (1990). A distributed model for mobile robot environment-learning and navigation. Technical Report 1228, Massachusetts Institute of Technology Artificial Intelligence Lab, Cambridge, Massachusetts.

Matheson, T. (1997). Hindleg targeting during scratching in the locust. *Journal of Experimental Biology*, 200:93–100.

McClelland, J. and Rumelhart, D. (1988). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press. two volumes.

McDougall, W. (1923). *Outline of Psychology*. Methuen, London.

McGurk, H. and MacDonald, J. (1976). Hearing lips and seeing voices. *Nature*, 264:746–748.

Minsky, M. (1985). *The Society of Mind*. Simon and Schuster Inc., New York, NY.

Myers, K. L. (1996). A procedural knowledge approach to task-level control. In *Proceedings of the Third International Conference on AI Planning Systems*.

Neely, J. H. (1991). Semantic priming effects in visual word recognition: A selective review of current findings and theories. In Besner, D. and Humphreys, G. W., editors, *Basic Processes in Reading: Visual Word Recognition*, chapter 9. Lawrence Erlbaum Associates.

Nelson, K. (1990). Hierarchical organization, revisited. *Netherlands Journal of Zoology*, 40(4):585–616.

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

Newell, A. and Simon, H. A. (1972). *Human Problem Solving*. Prentice-Hall.

Nilsson, N. (1984). Shakey the robot. Technical note 323, SRI International, Menlo Park, California.

Nilsson, N. (1994). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1:139–158.

Norman, D. A. and Shallice, T. (1986). Attention to action: Willed and automatic control of behavior. In R.Davidson, Schwartz, G., and Shapiro, D., editors, *Consciousness and Self Regulation: Advances in Research and Theory*, volume 4, pages 1–18. Plenum, New York.

Parker, L. E. (1998). ALLIANCE: An architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2).

Pebody, M. (1995). Learning and adaptivity: Enhancing reactive behaviour architectures in real-world interaction systems. In Moran, F., Moreno, A., Merelo, J., and Chacon, P., editors, *Advances in Artificial Life (Third European Conference on Artificial Life)*, pages 679–690, Berlin. Springer-Verlag.

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

Port, R. F. and van Gelder, T., editors (1995). *Mind as Motion: Explorations in the Dynamics of Cognition*. MIT Press, Cambridge, MA.

Reeves, R. and Hallam, J. C. (1995). Control of walking by central pattern generators. In *Intelligent Autonomous Systems Four (IAS95)*, Karlsruhe. also Edinburgh DAI RP-726.

Reilly, W. S. N. (1996). Believable social and emotional agents. Technical Report CMU-CS-96-138, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. Ph.D. thesis.

Rosenblatt, K. and Payton, D. (1989). A fine-grained alternative to the subsumption architecture for mobile robot control. In *Proceedings of the IEEE/INNS International Joint Conference on Neural Networks*.

Russell, S. J. and Norvig, P. (1995). *Artificial Intelligence : A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ.

Saltzman, E. L. (1995). Dynamics and coordinate systems in skiled sensorimotor activity. In Port, R. F. and van Gelder, T., editors, *Mind as Motion: Explorations in the Dynamics of Cognition*, chapter 4, pages 149–173. MIT Press, Cambridge, MA.

Sloman, A. and Logan, B. (1998). Architectures and tools for human-like agents. In *Proceedings of the Second European Conference on Cognitive Modelling*, pages 58–65, Nottingham, UK. University of Nottingham Press.

Steels, L. (1994). Building agents with autonomous behavior systems. In Steels, L. and Brooks, R., editors, *The 'artificial life' route to 'artificial intelligence'. Building situated embodied agents.* Lawrence Erlbaum Associates, New Haven.

Stone, P. and Veloso, M. (1999). Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*.

Thórisson, K. R. (1999). A mind model for multimodal communicative creatures & humanoids. *International Journal of Applied Artificial Intelligence*.

Thórisson, K. R. and Bryson, J. (*in preperation*). Flexible behavior-based planning for multimodal characters capable of task-oriented dialogue and action.

Tinbergen, N. (1951). *The Study of Instinct*. Clarendon Press, Oxford.

Townsend, J. T. and Busemeyer, J. (1995). Dynamic representation of decision-making. In Port, R. F. and van Gelder, T., editors, *Mind as Motion: Explorations in the Dynamics of Cognition*, chapter 4, pages 101–120. MIT Press, Cambridge, MA.

Tyrrell, T. (1993). *Computational Mechanisms for Action Selection*. PhD thesis, University of Edinburgh. Centre for Cognitive Science.

van Gelder, T. (1998). The dynamical hypothesis in cognitive science. *Behavioral and Brain Sciences*, 21(5):616–665.

Vereijken, B. and Whiting, H. T. A. (1998). Hoist by their own petard: The constraints of hierarchical models. *Brain and Behavioral Sciences*, 21(5):695.

Wilkins, D. E. and Myers, K. L. (1998). A multiagent planning architecture. In *Proceedings of AIPS-98*, pages 154–162.

Wilkins, D. E., Myers, K. L., Lowrance, J. D., and Wesley, L. P. (1995). Planning and reacting in uncertain and dynamic environments. *Journal of Experimental and Theoretical AI*, 7(1):197–227.

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2).