Real-time Control of Animated Broad Agents

A. Bryan Loyall and **Joseph Bates**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
bryan.loyall@cs.cmu.edu, joseph.bates@cs.cmu.edu

Abstract

As autonomous agents' interactions with humans become richer, we believe it will become increasingly important for some of the agents to have believable and engaging personalities. In previous papers we have described Tok, a broad agent architecture which integrates reactivity, goal-directed behavior, emotion and some memory and inference for agents in non-real-time worlds. In this paper we discuss the issues raised when we extend Tok to work in real-time, animated domains. Convincing animated motion poses three challenges to the architecture: multiple primitive actions and higher level activities must be executed simultaneously; future actions must be known before current actions complete, to enable smooth animation; and the mind must be fast enough to provide the impression of awareness. Here we describe Hap, the reactive substrate of Tok, and its approaches to these challenges. The described architecture was used for the creation of three agents, called woggles, in a world titled *Edge of Intention*, which was first shown at the AAAI-92 AI-based Arts Exhibition.

Introduction

Artists regularly construct characters when they create stories in noninteractive media such as animated films or novels.[1] There are known techniques for making the constructed characters believable and engaging with distinct, recognizable personalities (Thomas & Johnston, 1981; Gardner, 1991). Computer scientists also create characters in the form of autonomous agents for interactive worlds. As these agents' interactions with humans become richer, we believe it will become increasingly important for some of them to have believable and engaging personalities.

To this end we are studying requirements and techniques to construct believable agents for interactive worlds. We suspect one requirement for believable interactive agents is that they have broad though perhaps shallow capabilities. This requirement is described in (Bates, Loyall & Reilly, 1991), and our efforts toward constructing such broad agents have been described in (Bates, Loyall & Reilly, 1992b; Bates, Loyall & Reilly, 1992a). Those papers describe an agent architecture, called Tok, that integrates reactivity, goal-directed behavior, emotion and some memory and inference for agents in interactive fiction worlds.

In this paper we discuss the issues raised when extending this work to believable agents for a real-time, animated domain. We found three challenges in this extension. First, when an animated creature is performing a sequence of actions, the next action must often be known in advance to determine in detail how to animate the current action. For example, depending on whether the agent is immediately jumping again or stopping, the way the agent transfers momentum during the final part of a jump is different.

Second, creatures we normally see, whether real or animated, perform multiple actions at one time. Dogs wag their tail, and move their ears, head and eyes while they are barking to get your attention. If we want our creatures to have convincing behavior, they also need to be able to perform multiple actions and pursue multiple higher level activities concurrently.

Third, creatures in a real-time domain must think fast enough to keep up. In our domain primitive actions have durations between 100 and 1500 milliseconds. The creatures must be able to respond to their own and other creatures' actions as they are occurring. In addition, to appear active they must be able to produce actions at the same rate they are being completed by their own bodies. This demand is increased because multiple actions are often executing simultaneously.

Our approaches to these challenges largely appear in the reactive substrate of the Tok architecture, called *Hap*, so Hap will dominate our discussion. For information about Tok as a whole see (Bates, Loyall & Reilly, 1992b; Bates, Loyall & Reilly, 1992a). For detailed information about the emotion component of Tok see (Reilly & Bates, 1992).

The architecture described here was used for the creation of three agents, called woggles, in a world titled *Edge of Intention*. This world was first shown at the AAAI-92 AI-based Arts Exhibition, and is visiting other sites. The architecture is part of the Oz project at

[1]This paper originally appeared in the Proccedings of the Fifteenth Annual Conference of the Cognitive Science Society, Boulder, CO, June 1993.

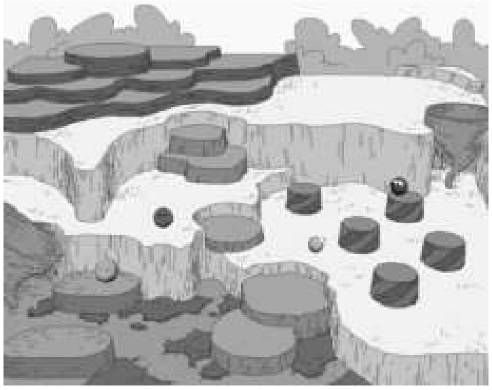

Figure 1: The *Edge of Intention* world

Carnegie Mellon, which develops technology for artistically interesting, highly interactive, simulated worlds. These worlds are intended to give users the experience of living in (not merely watching) dramatically rich worlds that include believable, engaging agents (Bates, 1992).

Domain

The woggle's world is shown in Figure 1. The physical world is designed to be something like a child's playground or one of the animal environments in a zoo. The style was intended to be reminiscent of the worlds created by Dr. Seuss in his popular children's books (Seuss, 1975). There are areas in the world which are conducive to sleeping, playing, exploring, exercising, showing off and observing the world. The three woggles use these areas as well as the rest of the world as they go about their normal behaviors. They play together, fight, relax, explore the world, mope when they are sad, try to cheer up their sad friends, etc. Figure 3 lists additional behaviors. The user can take part in many of these activities by controlling a fourth woggle, which is otherwise inactive, with the mouse.

The three creatures themselves were designed to have somewhat stereotypical personalities: Wolf is an aggressive character; Shrimp is a friendly, meek one; and Bear is a protector. These characterizations are reflected in the agents' behavior. Bear becomes sad when there is strife in the world and will often try to stop it, while Wolf sees it as an opportunity to amuse himself. Shrimp is often afraid, and almost never does anything aggressive.

Specifically, the world is modeled as a height field with simple physics. It is a rectangularly bounded plane with each point having an associated height representing the terrain surface at that point[2]. Each agent's body is an ellipsoid with eyes.

The agents cannot see the colors painted onto the world, but can sense the shape of the world by querying heights at individual points in the x,y plane. They

[2]The two funnel shaped chutes are actually cylinders with the funnel shape painted onto the surface. The bushes in the background are likewise painted.

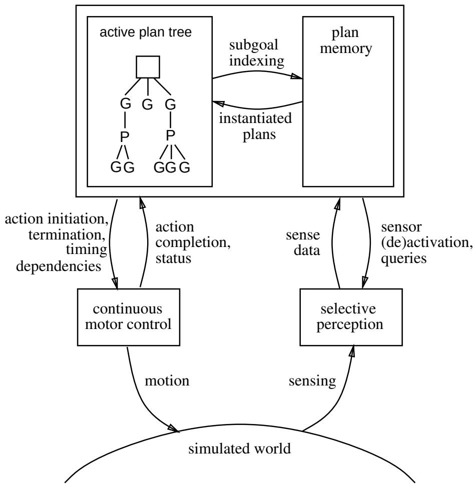

Figure 2: The Hap Architecture

can also sense each other to determine properties such as where an agent is located, where an agent's eyes are pointed, which primitive actions are being performed, etc. Active sonars are connected to the system so that the creatures can sense the presence and location of nearby humans.

In each creature, actions are sent from Tok to the agent's body to be executed. The body notifies the mind when actions are actually started and when they finish. Multiple actions can be executed simultaneously if they do not require the same body resources[3]. The primitive acts which can be performed are jump, turn, squash, puff up, slide, change color, move eyes and face a point or track it with eyes.

The task facing Tok is to produce a series of potentially overlapping actions that make the creature seem alive.

Basic Hap

Hap is the reactive substrate of the Tok architecture. In this section we describe the basic Hap architecture. An early version of basic Hap has been described in (Loyall & Bates, 1991). Basic Hap developed from our desire to extend the ideas of situated activity and reactivity (Agre & Chapman, 1990; Brooks, 1986) with explicit goals, which we felt necessary in broad agents. Hap has similarities with other reactive architectures, (Nilsson, 1992; Simmons, 1991; Maes, 1989; Kaelbling & Rosenschein, 1990), but is most similar to Firby's RAPs (Firby, 1989) and Georgeff and Lansky's PRS system (Georgeff & Lansky, 1987). Like several of these systems, Hap facilitates hierarchical composition of behaviors from primitive actions.

The Hap architecture is shown in Figure 2. Hap directly supports goal-directed action producing behaviors, and allows the encoding of cognitive tasks. It contin-

[3]Each woggle body has 27 resources including such things as the center of the woggle, which direction it is facing, etc.

uously chooses the agent's next action based on perception, current goals, emotional state and aspects of internal state. Goals in Hap contain an atomic name and a set of parameters which are instantiated when the goal becomes active, for example (tease <other>). Goals do not characterize world states to accomplish, and Hap does no planning. Instead, sets of actions (which for historical reasons we call "plans") are chosen from an unchanging plan library. These plans are either ordered or unordered collections of subgoals, physical actions and mental actions which can be used to accomplish the invoking goal. For example, one plan for the above tease goal is the sequence of subgoals: (goto <other>), (greet <other>), (run-from <other>). Subgoals are constructed by the author, primitive physical actions are given by the domain, and mental actions are given by the underlying language (either Lisp or C), and typically include such actions as storing a fact, performing calculations, etc.

Plans have testable *preconditions* which are true when the plan is applicable for a goal in the current state of the world. Multiple plans can be written for a given goal, with Hap choosing between the plans at execution time. If a plan fails, Hap will attempt any alternate plans for the given goal, and thus perform a kind of backtracking search in the real world.

Hap stores all active goals and plans in a structure called the active plan tree (APT). This is a tree of alternating layers of goals and plans. The first layer of the APT is the collection of top level goals for the creature. A goal's child, if it has one, is the active plan for that goal. A plan's children are its component subgoals, physical actions and mental actions. The APT is constantly changing: expanding as plans with their component subgoals are chosen for goals and shrinking as goals and plans succeed and fail. Physical actions succeed or fail depending on their realization in the world. Mental actions always succeed. Goals succeed when a plan for the goal succeeds, and fail if all of the applicable plans have failed. Plans succeed if all of the component steps succeed and fail if any of the steps fail.

There are various annotations in the APT to support reactivity and the management of multiple top-level goals. Two important annotations are *context conditions* and *success tests*. Both of these are arbitrary testable expressions over the perceived state of the world and other aspects of internal state. Success tests can be associated with any goal in the APT. When a success test is true, its associated goal is deemed to have been accomplished and thus no longer needs to be pursued. For example, the first step of the tease plan described above has a success test associated with it to determine if the agent is already near <other>. If this test is true when the plan begins, the step (goto <other>) would be skipped. Also, if the agent is in the process of going toward <other> when some external factor causes the test to be true, the success test would enable Hap to recognize that the goal has succeeded and stop pursuing it.

Analogously, context conditions can be associated with plans in the active plan tree. When a context condition becomes false its associated plan is deemed no longer applicable in the current state of the world. That plan fails and a new plan must be chosen to accomplish the invoking goal.

Every goal instance has a *priority* number, used by Hap when choosing a goal to execute, and an *importance* number, used by the emotion system when considering the significance of the goal. These annotations are assigned to instances of goals rather than to types of goals, because identical goals could have different priority or emotional importance depending on the context in which they arise. For example, the goal of going to an area as a subgoal to a run away goal would likely have a higher priority than the same goal in pursuit of an exploration goal.

Any goal can be annotated as persistent with respect to success, failure or both. A goal so marked remains in the tree upon success or failure rather than being removed, and is reset so that it is again available to be pursued.

Multiple plans for a goal can be partially ordered by specificity using numeric annotations. Hap uses these to choose more specific plans when multiple plans apply.

Hap executes by first modifying the APT based on changes in the world: goals whose success tests are true and plans whose context conditions are false are removed along with any subordinate subgoals or plans. Next, one of the leaf goals is chosen. This choice is made by a goal arbiter which prefers high priority goals and, among goals of equal priority, prefers continuing the active line of expansion. If the chosen goal is a primitive mental action, it is executed. If it is a physical action, it is sent to the body to be executed. Otherwise, the plan library is indexed and the plan arbiter chooses one plan for the goal from among those whose preconditions are true. The plan arbiter will not choose plans which have already failed to achieve this goal instance, and prefers more specific plans over less specific ones using the specificity annotation. The chosen plan and its component subgoals are added to the APT, and the execution loop repeats.

Hap includes a special type of goal called *wait* that is never chosen by the goal arbiter. Thus, when present in a sequential plan it suspends that plan until removed. It can be removed by an associated success test becoming true, or by success or failure of one of its ancestors in the tree. Arbitrary demons can be encoded using wait goals.

Goals can have two additional annotations: ignore-failure and effect-only. Goals marked with ignore-failure treat failure as success thus making the attempt of the goal enough for success. The effect-only annotation causes the marked goal to be irrelevant in determining the success of its parent plan. The parent plan then succeeds when all of its other subgoals succeed regardless of whether the marked goal has been attempted.

Hap as described in this section is implemented in Lisp and is currently being used to develop agents for the Oz project's text interface interactive fiction worlds.

Real-Time Hap

To respond to the challenges of a real-time, animated domain, a number of refinements to this basic architecture are necessary: (1) parallel execution of multiple actions and goals, (2) early production of next actions to allow smooth animation, (3) automatic management of selective sensors, and (4) incremental evaluation of the continuously monitored conditions.

Parallel Execution of Goals and Actions

As mentioned earlier, for agents to appear realistic, they must do more than one thing at a time. Basic Hap allows agents to hold multiple parallel goals through the top level parallel set of goals and through parallel plans which arise during execution. Like other goal-directed reactive architectures (Firby, 1989; Georgeff & Lansky, 1987), basic Hap manages these multiple goals by concentrating on the most critical according to its arbitration mechanism, and for the most part only attends to other goals after the current goal completes or as events trigger demons.

In real-time Hap, all of an agent's active goals can be attended to, potentially producing multiple actions or performing parallel cognitive processing. Hap uses a greedy approach by attending to the most critical goals first and mixing in others as time allows. In each decision cycle Hap chooses the most critical of the available leaf goals. This thread of behavior is attended to until it is interrupted, as in basic Hap, or it becomes suspended. For example, when a jump action in a sequential plan is sent to the body to be executed, the plan cannot continue until the action completes. When a thread is suspended Hap uses the available processing time (in this case approximately 1200 milliseconds of real time) to attend to the other, perhaps unrelated, available goals. A thread of behavior could also be suspended if its current goal is the special form *wait* or if its current goal is incompatible with a more critical executing thread.

This notion of incompatible goals deserves further comment. Two actions are considered incompatible if they use the same body resources. For example the jump and slide actions both move the body and so cannot execute simultaneously. Similarly, goals can be incompatible with other goals or with actions. These goal incompatibilities exist independently of any primitive action resource conflict. For example, while sleeping the primitive actions being executed are rather sparse and do not by themselves preclude concurrently executing other primitive actions. In the woggles, however, the goal of sleeping is incompatible with actions or goals which significantly move the body.

Hap allows authors to specify pairs of goals or actions which are incompatible. The woggles have an average of 49 pairs each. During processing, Hap will not allow two incompatible goals to be pursued at the same time. The more critical one according to the goal arbiter is pursued while the other is temporarily suspended.

Early Production of Next Action

To allow the motor control part of the animation system to provide smooth motion, Hap attempts to provide the next action for each thread before the current action finishes. One hundred milliseconds prior to the completion of an action Hap assumes that the action will complete successfully. It then can use this time to compute the next action along that thread. If an action is produced, it is sent to the motor system to be executed after the current action. All of Hap's reactive mechanisms apply to these pending actions as well as to normal Hap execution, so in the event that Hap chooses to abort a pending action, a message is sent to the motor system and it is removed.

Of course, if the agent is currently attending to something more critical than this thread, it will continue to be attended to and the next action will likely not be computed. The motor control system will assume that action for this set of muscles is temporarily ending. Also, if the current action fails after it has been assumed to finish successfully, the agent must recover from its incorrect assumption using its various reactive mechanisms, such as success tests and context conditions.

Selective Sensing

Sensing in a real-time, animated world must be efficient. To this end, Hap creatures employ task-specific sensors which can be turned on or off as needed. Each sensor observes a low level aspect of the world and notifies the mind when that aspect's value changes. Typical sensors are "can I see woggle X jumping" and "what is the position of woggle X". The aspects of the world which must be known to evaluate an agent's preconditions, success tests and context conditions are noted when these conditions are written by associating a list of sensors for each condition. Hap automatically manages the sensors by turning them on and off when appropriate. As a leaf subgoal is chosen to be executed, sensors needed to evaluate the preconditions for that goal's plans are automatically turned on, and then turned off again after a plan is chosen. Likewise, when a particular goal or plan is present in the APT, the sensors relevant to evaluating any success tests or context conditions are turned on. When that goal or plan is removed from the tree because of success, failure or irrelevance, the sensors are turned off. Because the same sensor may be needed for several different conditions at a time, the sensors are shared and reference counted. In the woggles we observed roughly a factor of two reduction in active sensors as a result of this sharing.

Incremental Evaluation of Conditions

Typically a Hap agent has a number of continuously monitored conditions (context conditions and success tests) active at any given time. In order for these agents to run fast enough for a real-time animation system, we believe it is useful to evaluate them incrementally. Thus characters written in the Hap language are compiled to RAL (Forgy, 1991), a production system extension of the C programming language that includes a RETE (Forgy,

Action Behaviors		Emotion Behaviors
follow-the-leader	save-friend	handle-goal-failure
lead-the-follower	seek-out-woggle	handle-goal-success
freak-out	sleep-until-rested	infer-blame
go-to-place	stay-in-view	infer-credit
go-to-woggle	release-aggression	fear-being-hurt
help-friend	tease-woggle	react-to-threat
jump-in-chute	watch-human	react-to-threat-to-other
mope	dance-on-pedestals	react-to-falling-behind-in-follow-the-leader
watch-nervously	tremble	
rest	dart-eyes	clean-up
roll-eyes	watch-a-woggle	decay-emotions
run-away	look-for-human	
sigh	blink	Sensing Behaviors
console-friend	threaten	recognize-aggression
do-fun-acrobatics	fight-back	recognize-escape
greet-human	protect-friend	recognize-hey
hey	gang-up-on	recognize-threat-to-other
wander-explore	amuse	recognize-teasing
threaten-for-fun		recognize-moping

Figure 3: Woggle Behaviors

1982) incremental matcher implementation. The character's APT is represented in working memory, with context conditions, success tests, and preconditions compiled to rules. These rules fire to prune and expand the tree, with additional runtime support rules included to complete the architecture. In *Edge of Intention*, the real-time Hap agents execute approximately 50 times faster than basic Hap agents.

Building Higher Functionality on Hap

Hap was initially designed to be a reactive architecture for action. As we have continued to build creatures, its role has broadened somewhat. In addition to the action producing behaviors, aspects of sensing and emotion are implemented using Hap behaviors. Some of the behaviors for the woggles are listed in Figure 3. (A complete woggle contains about 250 goal types and 500 plans.)

The task-specific sensors provide sensory information at a low level, including primitive action events. To be effective, agents need to be able to recognize abstract composites of this sensory information such as agents fighting, playing games or moping. Originally we envisioned constructing new task-specific sensors for each of these higher level events, but we realized that Hap's normal mechanisms could conveniently be used to create recognizers using only the low-level sensors. The constructed behaviors use parallel and sequential plans, combined with success tests, context conditions, mental actions, etc. to actively look for patterns in the low level perceptions as they are perceived. For example, a sensing behavior might infer that a woggle is trying to annoy another because over time the first is staying very close and performing quick, jerky actions, even though the second is trying to move away.

Tok's emotion architecture, Em (Reilly & Bates, 1992), is also implemented using Hap behaviors. Em creates emotions by matching particular patterns of events in the world with the agent's goal state. Hap provides flexible access to the goal state because it is represented in working memory. Em is implemented by combining this reflection of the goal state with abstract event recognition behaviors as above to recognize events such as another agent causing this agent's goal to fail which then causes anger. Em also periodically decays its emotion state, and performs various internal clean-up tasks using Hap behaviors.

By implementing these capabilities as Hap behaviors, they inherit the processing properties of the architecture. They are managed as some of the multiple threads of behavior, with Hap attending to several of them simultaneously when the time available to think allows, or otherwise only attending to the ones most critical to this agent.

Example of Processing

To better illustrate the flow of Hap processing, we present here a brief excerpt of one creature's processing, with special attention to the concepts we have described above. The full excerpt lasts roughly one second of real-time in the running system.

At the beginning of the excerpt Bear is pursuing his amuse goal by going toward a hill in the world to watch the other woggles. He has sent the next jump action to his body and is waiting for it to finish. At this point he notices the possibility that the user-controlled woggle is intimidating Shrimp. He notices this because the user is close to Shrimp and has just performed a quick movement. This combination of sensations is recognized by the firing of a success test in one of his persistent sensing behaviors. This behavior is higher priority than the amuse thread so it would interrupt in any case, but since it does not conflict and the amuse behavior is waiting for the action to finish, it is pursued concurrently. As it is pursued, it verifies whether the user is in fact threatening Shrimp by watching their actions unfold. If the behavior were to determine that no threatening is taking place, it would be aborted by a context condition firing, and the persistent goal would be reset so that it is ready to notice such a situation again. In this instance, however, the behavior observes the user slide in front of Shrimp's face again and puff up. This is enough to recognize that the user is intimidating Shrimp.

This recognition gives rise to a number of goals: two emotion producing goals and a goal to protect Shrimp. The protect goal conflicts with the existing amuse goal and has a higher priority, so the amuse goal and its subgoals are temporarily suspended. The jump action itself cannot be aborted as Bear is flying through the air, so it is allowed to complete normally. When the protect behavior is elaborated and a jump toward the fight is chosen as the next act, it conflicts with this orphaned jump, but again has a higher priority. Since the act cannot be aborted, the next jump is queued after it enabling the motor system to create an appropriate landing for the current jump. After this decision, Hap again has time to mix in other behaviors. The two emotion goals are processed resulting in Bear becoming sad because there is a fight in the world and angry at the user for picking on a friend of his. The anger then increases his *energy* behavioral feature which affects how he pursues his behaviors, such as the speed of jumping toward the fight.

Conclusion

We have described the Hap architecture including extensions for convincing, real-time, animated motion. We believe a number of qualities of the architecture have contributed to our attempts to create believable agents. These include improving the speed of the architecture, providing task-specific sensing, permitting multiple actions and goals to be pursued concurrently, and providing early production of actions to enable smooth animation. In addition, Hap provides a common computational environment for other parts of the Tok architecture, namely sensing and emotion, scheduling them along with other goals of the agent.

This architecture has been used to create three animated creatures which were first shown at the AAAI-92 AI-based Arts Exhibition in the exhibit *Edge of Intention*. The system runs at 10 frames per second on a Silicon Graphics Indigo XS24+Z with an R3000 processor. Each mind uses approximately 20 milliseconds of each 100 millisecond video frame for its processing; the remainder is used for graphics.

Precisely evaluating the degree to which we have succeeded in our goal of building believable, engaging characters is of course problematic. Ultimately, believability must be a subjective judgment based on observations of and interactions with the constructed creatures. At the time of this writing several thousand people have seen *Edge of Intention* and anecdotal evidence suggests that we have made some progress toward believability. The system has been chosen for display at the curated SIGGraph 1993 Art Show and a the Carnegie Mellon Hewlett art gallery. It is on display in the Boston Computer Museum's permanent collection, and will be shown in the ARS Electronica '93 exhibition in Austria.

The specific behaviors, emotions and personalities of the woggles were created in less than six weeks of work. We suspect that further artistic effort working within existing Tok will yield correspondingly more believable creatures. Work is continuing to extend Tok in the areas of emotion, models of other agents, integration of natural language understanding and generation including pragmatics and emotional speech, and Hap is evolving to support new demands of these systems.

Acknowledgments

This research was supported in part by Fujitsu Laboratories, Ltd. We also thank those who helped construct *Edge of Intention*: A. Witkin, J. Altucher, A. Hauptmann, M. Kantrowitz, D. Langer, K. Murakami, P. Olbrich, Z. Popovic, W.S. Reilly, P. Sengers, W. Welch, P. Weyhrauch, and Production Systems Technologies, Inc.

References

Agre, P. E. & Chapman, D. 1990. What are plans for? In *Robotics and Autonomous Systems*. Elsevier Science Publishers.

Bates, J. 1992. Virtual reality, art, and entertainment. *PRESENCE: Teleoperators and Virtual Environments*, 1(1):133–138.

Bates, J., Loyall, A. B., & Reilly, W. S. 1991. Broad agents. In *Proceedings of AAAI Spring Symposium on Integrated Intelligent Architectures*, Stanford, CA. Available in *SIGART Bulletin*, Volume 2, Number 4, August 1991, pp. 38-40.

Bates, J., Loyall, A. B., & Reilly, W. S. 1992a. An architecture for action, emotion, and social behavior. In *Proceedings of the Fourth European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, S.Martino al Cimino, Italy.

Bates, J., Loyall, A. B., & Reilly, W. S. 1992b. Integrating reactivity, goals, and emotion in a broad agent. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, Bloomington, IN.

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

Firby, J. R. 1989. *Adaptive Execution in Complex Dynamic Worlds*. PhD thesis, Department of Computer Science, Yale University.

Forgy, C. L. 1982. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1):17–37.

Forgy, C. L. 1991. *Rule-extended Algorithmic Language Language Guide*. Production Systems Technologies, Inc., 5001 Baum Boulevard, Pittsburgh, PA 15213.

Gardner, J. 1991. *The art of fiction : notes on craft for young writers*. Vintage Books, New York, vintage books edition.

Georgeff, M. P. & Lansky, A. L. 1987. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*.

Kaelbling, L. P. & Rosenschein, S. J. 1990. Action and planning in embedded agents. *Robotics and Autonomous Systems*, 6(1-2):35–48.

Loyall, A. B. & Bates, J. 1991. Hap: A reactive, adaptive architecture for agents. Technical Report CMU-CS-91-147, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Maes, P. 1989. The dynamics of action selection. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI.

Nilsson, N. J. 1992. Toward agent programs with circuit semantics. Technical Report STAN-CS-92-1412, Department of Computer Science, Stanford University, Stanford, CA.

Reilly, W. S. & Bates, J. 1992. Building emotional agents. Technical Report CMU-CS-92-143, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Seuss, D. 1975. *Oh, the THINKS You Can Think!* Random House, Inc., New York.

Simmons, R. 1991. Concurrent planning and execution for a walking robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Sacramento, CA.

Thomas, F. & Johnston, O. 1981. *Disney Animation – The Illusion of Life*. Abbeville Press, New York.