Integrating Reactivity, Goals, and Emotion in a Broad Agent

Joseph Bates A. Bryan Loyall W. Scott Reilly

May 1992
CMU-CS-92-142

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

To appear in the Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society, Bloomington, Indiana, July 1992.

Keywords: artificial intelligence, agent architecture, autonomous agents, emotion, goals, reactivity, interactive media, virtual reality, Oz

Abstract

Researchers studying autonomous agents are increasingly examining the problem of integrating multiple capabilities into single agents. The Oz project is developing technology for dramatic, interactive, simulated worlds. One requirement of such worlds is the presence of broad, though perhaps shallow, agents. To support our needs, we are developing an agent architecture, called Tok, that displays reactivity, goal-directed behavior, and emotion, along with other capabilities.

Integrating the components of Tok into a coherent whole raises issues of how the parts interact, and seems to place constraints on the nature of each component. Here we describe briefly the integration issues we have encountered in building a particular Tok agent (Lyotard the cat), note their impact on the architecture, and suggest that modeling emotion, in particular, may constrain the design of integrated agent architectures.

1 Broad Agents

The Oz project [3] at Carnegie Mellon is developing technology for artistically interesting, highly interactive, simulated worlds. We want to give users the experience of living in (not merely watching) dramatically rich worlds that include moderately competent, emotional agents.

An Oz world has four primary components. There is a simulated physical environment, a set of automated agents which help populate the world, a user interface to allow one or more people to participate in the world [13], and a two-player adversary search planner concerned with the long term structure of the user's experience [2]. Oz shares some goals with traditional story generation systems [18, 16], but adds the significant requirement of rich interactivity.

One of the keys to an artistically engaging experience is for the user to be able to "suspend disbelief". That is, the user must be able to imagine that the world portrayed is real, without being jarred out of this belief by the world's behavior. The automated agents, in particular, mustn't be blatantly unreal. Thus, part of our effort is aimed at producing agents with a broad set of capabilities, including goal-directed reactive behavior, emotional state and behavior, and some natural language abilities. For our purpose, each of these capacities may be as shallow as necessary to allow us to build broad, integrated agents [4].

Oz worlds are far simpler than the real world, but they must retain sufficient complexity to serve as interesting artistic vehicles. The complexity level is somewhat higher, but not exceptionally higher, than typical AI micro-worlds. Despite these simplifications, we find that our agents must deal with imprecise and erroneous perceptions, with the need to respond rapidly, and with a general inability to fully model the agent-rich world they inhabit. We suspect that some of our experience with broad agents in Oz may transfer to other domains, such as social, real-world robots.

Building broad agents is a little studied area. Much work has been done on building reactive systems [6, 10, 9, 24], natural language systems, and even emotion systems [8, 23, 21]. There is growing interest in integrating action and learning (see [14]) and some very interesting work on broader integration [25, 22]. However, we are aware of no other efforts to integrate the particularly wide range of capabilities needed in the Oz domain. Here we present our efforts, focusing on integration mechanisms and their impact on components of the architecture.

2 Tok and Lyotard

In analyzing our task domain, we concluded that the capabilities needed in our initial agents are perception, reactivity, goal-directed behavior, emotion, social behavior, natural language analysis, and natural language generation. Our agent architecture,

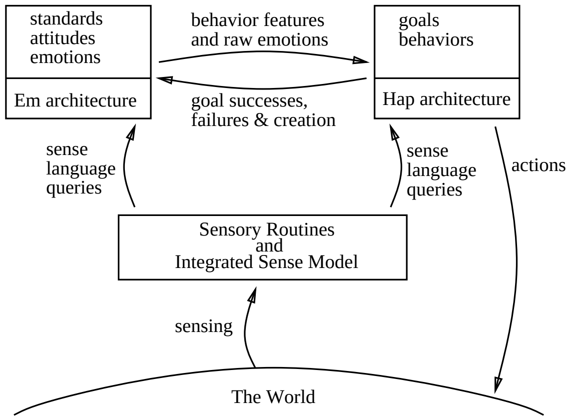

Figure 1: Tok Architecture

Tok, partially (but not fully) partitions these tasks into several communicating components. Low-level perception is handled by the Sensory Routines and the Integrated Sense Model. Reactivity and goal-directed behavior are handled by Hap [17]. Emotion and social relationships are the domain of Em [?]. Language analysis and generation are performed by Gump and Glinda [12, 13]. Figure 1 shows how these components, excluding Glinda and Gump, are connected to form Tok.

In the remainder of this section we describe the components of Tok in just enough detail to allow discussion of the integration issues. For a more complete description see [5]. We illustrate the description using examples from an existing Tok agent, a simulated house cat named "Lyotard", which exercises most of the capabilities of the architecture.

Table 1 lists the emotions and behaviors from our original informal design document for Lyotard. Our goal in developing Lyotard was to build a creature that could believably pass for a cat in an Oz micro-world. The emotions shown are those naturally available in the current version of Em, though in the end we did not use all of them. The behaviors were developed over several hours by the cat owners in our group. The behavioral features are used to modify the style of particular behaviors. They are usually derived from Lyotard's emotional state, though they also can be directly adjusted by behaviors.

2.1 The Simulated World and Perception

The Oz physical world is an object-oriented simulation. Agents sense the world via sense data objects which propagate from the item sensed through the world to the agents. Each sense datum describes the thing sensed as a collection of property/value pairs. Unique names are not used to identify objects; agents must derive identity from

Emotions	Behaviors		Features
hope†	wanting to be pet	purring	curious
fear	chasing ball/*creatures*	arch back	content
happy	wanting to go out/in	hiss	aggressive
sad	*pouncing on creatures*	swat	ignoring
pride	wanting to eat	bite	friendly
shame	getting object	run away	*proud*
admiration	(using human)	have fun	energetic
reproach	searching for thing	eating	
gratification	cleaning self	*crazy hour*	
remorse	playing with ball	rubbing	
gratitude	playing with mouse	licking	
anger	*carrying mouse*	watching	
love	hiding (anger/fear)	sit in sun	
hate	pushing things around		

†*italicized items were not included in final implementation*

Table 1: Original Lyotard Task

other properties. Sense data can be transformed as they travel. For example speech behind a closed door can be muffled. In general, the sense data available to an agent can be incomplete, incorrect, or absent. Agents perform actions by invoking methods on appropriate sets of objects. These methods may alter the world, propagate sense data, and succeed or fail.

Each Tok agent runs by executing a three step loop: sense, think, act. During each sense phase a snapshot of the perceivable world is sensed and the data is recorded in the sensory routines. These snapshots are time-stamped and retained. An attempt is then made to merge them into the Integrated Sense Model (ISM), which maintains the agent's best guess about the physical structure of the whole world. The continuously updated information in the sensory routines and the longer term, approximate model in the ISM are routinely queried when choosing actions or updating the emotional state of Lyotard.

2.2 Action (Hap)

Hap is Tok's goal-directed, reactive action engine [17]. It continuously chooses the agent's next action based on perception, current goals, emotional state, behavioral features and other 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 `(goto <object>)`. Goals do not characterize world states to accomplish, and Hap does no explicit planning. Instead, sets of actions (which we nonetheless call "plans") are chosen from an unchanging plan library which may contain one or more plans for each goal. These plans are either ordered or unordered collections of subgoals and actions which can be used to accomplish the invoking goal. Multiple plans can be written for a given goal, distinguished in part by a testable precondition. 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 hierarchical structure called the active plan tree (APT). There are various annotations in the APT to support reactivity and the management of multiple 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 may be associated with selected goals 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. This can happen before the goal is attempted in which case it is skipped or can happen during execution of the goal in which case it is aborted.

Similarly, context conditions may be associated with APT plans. 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 is removed from the tree along with any executing subgoals. The parent goal then chooses a new plan or fails.

Hap executes by first modifying the APT based on changes in the world. Goals whose success test is true and plans whose context condition is false are removed along with any subordinate subgoals or plans. Next one of the leaf goals is chosen. If the chosen goal is a primitive action, it is executed. Otherwise, the plan library is indexed and the plan arbiter chooses a plan for this goal from among those whose preconditions are true. The plan arbiter will not choose plans which have already failed, and prefers more specific plans over less specific ones. At this point the execution loop repeats.

2.3 Emotion (Em)

Em models selected emotional and social aspects of the agent. It is based on ideas of Ortony et al. [23]. Like that work, Em develops emotions from a cognitive base: external events are compared with goals, actions are compared with standards, and objects are compared with attitudes. Most of Em's possible emotions are shown in Table 1. We describe them very briefly here, but see [?] for details.

Happiness and sadness occur when the agent's goals succeed or fail. Hope and fear occur when Em believes that there is some chance of a goal succeeding or failing.

Pride, shame, reproach, and admiration arise when an action is either approved or disapproved. These judgments are made according to the agent's *standards*, which represent moral beliefs and personal standards of performance. Pride and shame occur when the agent itself performs the action; admiration and reproach develop in response to others' actions.

Some emotions are combinations of other emotions. Lyotard doesn't like to be touched when he's in the wrong mood and doing so will cause him sadness and reproach. These give rise to the composite emotion of anger at whomever pet him. Similarly, gratitude is a composite of happiness and admiration, remorse is sadness and shame, and gratification is happiness and pride.

Finally, love and hate arise from noticing objects toward which the agent has positive or negative *attitudes*. In Lyotard we use attitudes to model the human-cat social relationship. Lyotard initially dislikes the user, a negative attitude, and this attitude varies as the user does things to make Lyotard angry or grateful. As this attitude changes, so may his resulting love or hate emotions.

Emotions fade with time, but attitudes and standards are fairly stable. An agent will feel love when close to someone liked. This fades if the other agent leaves, but the attitude toward that agent remains relatively stable.

3 Component Integration

The arcs in Figure 1 denote the communication paths between the parts of Tok. The main interactions are Hap and Em querying perceptual information, Hap querying both the emotional state and behavioral features derived from the emotional state, and Em receiving notification of the creation, failure, and success of goals. Here we discuss only the communication between Hap and Em.

3.1 Hap's Communication with Em

As Hap runs, its active plan tree changes. These changes include goal creation and goal removal due to success or failure. As these events occur, Hap informs Em of what goals were affected, how they were affected, and the degree of importance that the agent builder associated with each of the goals. Em then uses this information to generate many of its emotions, as described above.

3.2 Behavioral Features

When we began the design of Lyotard, we expected that Hap would directly query Em's state. However, we found that Hap's decisions were often based on complex tests of the emotional state, and that the same tests arose repeatedly. Further, we sometimes wanted to produce behavior as if Em held a certain emotion which in fact was absent. It became clear that Lyotard's emotion-related behavior depended on an abstraction of the emotional state.

Lyotard:
(*go-to "closet")
(*lookaround "closet")
(*lookaround "closet")
(*jump-on "plant")
(*lookaround "plant")
(*walk-along "plant")
(*nibble "plant")
(*walk-along "plant")
(*jump-off "plant")
(*go-to "bedroom")
(*go-to "sunroom")
(*go-to "spare room")
(*jump-on "chair")
(*sit-down)
(*lick "lyotard")
(*lick "lyotard")
Player:
(*go-to "spare room")
L: (*jump-off "chair")
(*run-to "sunroom")
P: (*go-to "sunroom")
L: (*lookaround nervously)
P: (*pet "lyotard")
L: (*bite "player")
(*run-to "diningroom")
P: (*go-to "spare room")
L: (*lookaround nervously)
(*go-to "sunroom")
(*pounce-on "superball")
(*lookat "superball")
(*nudge "superball")
(*pounce-on "superball")
(*pounce-on "superball")
(*go-to "diningroom")
(*go-to "kitchen")
(*meow)
P: (*go-to "sunroom")
L: (*meow)
P: (*go-to "diningroom")
L: (*wait)
P: (*take "glass jar")
L: (*go-to "diningroom")
P: (*go-to "kitchen")
L: (*jump-on "table")
(*jump-off "table")
(*go-to "kitchen")
(*meow)
P: (*pour "jar" in "bowl")
L: (*eat "sardine")
(*eat "sardine")
(*eat "sardine")
(*eat "sardine")
(*eat "sardine")
P: (*pet "lyotard")
L: (*close-eyes lazily)
P: (*take "lyotard")
L: (*close-eyes lazily)

Figure 2: Section of an interaction with Lyotard

The abstraction, called "behavioral features", consists of a set of named features that modulate the activity of Hap. Features are adjusted by Hap or Em to control how Hap achieves its goals. Em adjusts the features to express emotional influences on behavior. It continuously evaluates a set of functions that determine certain features based on the agent's emotional state. Hap modifies the features when it wants to force a style of action. For example, it may decide to act friendly to help achieve a goal, even if the agent isn't feeling especially friendly.

Features may influence several aspects of Hap's execution. They may trigger demons that create new top-level goals. They may occur in the preconditions, success tests, and context conditions of plans, and so influence how Hap chooses to achieve its goals. Finally, they may affect the precise style in which an action is performed.

Table 1 shows Lyotard's behavioral features. The "aggressive" feature, for example, arises when Lyotard is either angry or mildly afraid (which might be considered bravado). This feature may affect Hap by giving rise to a new goal, such as bite-human, by influencing the choice of plan for a goal, such as nipping instead of meowing to attract attention, or by modifying the style of an action, such as swatting a toy mouse with unusual emphasis.

We have no structured set of features, and know of no source that suggests one. Besides those in Lyotard, we have seen the following suggested: curious, belligerent, persistent, depressed, patient [7]; timid, reckless, quiet, arrogant [11].

The feature mechanism, while very *ad hoc*, appears to provide a useful degree of abstraction in the interface between emotion and behavior. It is not merely a mechanism to vary Tok's behavior and thereby possibly increase the appearance of richness. Rather, it is an initial solution to the integration problem of driving behavior from both goals and emotion.

3.3 Results of Integration in Lyotard

We discuss here the beginning portion of a fragment of behavior that Lyotard has exhibited. The complete fragment is given in figure 2, and a correspondingly complete discussion can be found in [5]. The purpose of the trace is not to show the breadth of Lyotard's capabilities, which are better indicated by Table 1, but to demonstrate the integration of Tok and Em.

As the trace begins, Lyotard is engaged in exploration behavior in an attempt to satisfy a goal to amuse himself. (The explore behavior was not in the original Lyotard design presented in Table 1, but was added to Lyotard at a later stage). This behavior leads Lyotard to look around the room, jump on a potted plant, nibble the plant, etc.

After sufficient exploration, Lyotard's goal is satisfied. This success is passed on to Em which makes Lyotard mildly happy. The happy emotion leads to the "content" feature being set. Hap then notices this feature being active and decides to pursue a behavior to find a comfortable place to sit, again to satisfy the high-level amusement goal. This behavior consists of going to a bedroom, jumping onto a chair, sitting down, and licking himself for a while.

At this point, a human user whom Lyotard dislikes walks into the room. The dislike attitude, part of the human-cat social relationship in Em, gives rise to an emotion of mild hate toward the user. Further, Em notices that some of Lyotard's goals, such as not-being-hurt, are threatened by the disliked user's proximity. This prospect of a goal failure generates fear in Lyotard. The fear and hate combine to generate a strong "aggressive" feature and diminish the previous "content" feature. In this case, Hap also has access to the fear emotion itself to determine why Lyotard is feeling aggressive. All this combines in Hap to give rise to an avoid-harm goal and its subsidiary escape/run-away behavior that leads Lyotard to jump off the chair and run out of the room. (Space restrictions forbid us from continuing our discussion beyond this point, but see [5].)

The Oz system is written in Common Lisp and CLOS. Of the 50,000 lines of code that comprise Oz, the Tok architecture is roughly 7500 lines. Lyotard is an additional 2000 lines of code. On an HP Snake (55 MIPS), each Tok agent takes roughly two seconds for processing between acts. (Most of this time is spent sensing, which suggests that even in the interactive fiction domain it may be desirable to use task specific selective perception.)

4 Discussion of Tok and Related Work

Developing Tok forced us to consider several issues which may be of general interest: the requirements emotion may place on reactive architectures, using goals without building world models, producing coherent overall behavior from independent particular behaviors, and modeling personality and its influence on behavior.

4.1 Emotion, Explicit Goals, and World Models

Some researchers have argued in recent years that representing goals explicitly in agents presents serious obstacles to the production of robust, reactive behavior [6, 1]. Others, of course, disagree with this view and feel that goals are necessary to organize action (for instance, see many of the papers in [14] and the varied work on Soar [22]).

It is essential that Oz agents be reactive, so we have been sympathetic to the reactivity arguments made by Brooks, Agre, and others. However, it is also necessary for our agents to at least appear clearly to have goals and for them to exhibit emotion in response to events affecting those goals. This latter requirement, in particular, seemed unsolvable to us without explicitly representing goals within the agent.

Once we accepted the importance of reactivity and grounding in sensory inputs, which was forced upon us by facing our task squarely, it was not difficult to develop an architecture that represented goals explicitly while retaining reactivity. We were not forced to adopt the view of operators as pre/post condition pairs and goals as predicates on world states. It was not even necessary to view goals as testable expressions. Rather, we could view them simply as internalized tokens (perhaps with arguments) that guide Hap in choosing appropriate behaviors. Those behaviors may in turn contain other tokens, and so on.

Thus, we suggest that robust, reactive behavior is not diminished by the presence of explicit goals in an agent, but by the attempt to model the agent's choice of action as a planning process over characterizations of the world. Our view of goals allows us to avoid many of the unpleasant consequences of trying to model the world, while preserving the strengths of goals as a mechanism for organizing action. (Though we note that it may well be possible to combine these views in "plan-and-compile" architectures [20, 19], of which Soar is a particularly rich example [15].)

4.2 Mixing Independent Behaviors

As we have used the word, a behavior is a cluster of related goals and plans that produces some recognizable, internally coherent pattern of action. A behavior is often represented by a single high-level goal.

We initially developed the notion of a behavior to allow us to specify Lyotard. We needed some concise way to represent the major components of Lyotard's action, and attaching suggestive names to a set of high-level goals seemed helpful.

The goals were implemented independently, resulting in a set of independent behaviors. Each of these behaviors is composed of a set of Hap plans and subgoals, with appropriate context conditions and success tests.

The context conditions and success tests were developed to make each behavior robust in the face of changes in the world, be they unexpected failures or serendipitous success. We expected these surprises to be due to external events performed by other agents or unforeseen complexities in the physical nature of the world. However, it has turned out that the agent's own actions, performed by other interleaved behaviors, are one of the main causes of unexpected changes. The context conditions and success tests allow these independent behaviors to mix together fairly well, without much explicit design effort to consider the interactions. Thus, adding reactivity to goal-directed behavior seems to help support the production of coherent, robust overall behavior from independently executing particular behaviors.

4.3 Modeling Personality

Tok must support construction of a variety of agents by the artists building Oz worlds. A key facet of this support is allowing different personalities to be modeled without requiring that every agent be built from scratch. The behavioral feature mechanism appears to provide a simple means to help achieve this.

One can build behaviors to respond to a standardized set of behavioral features, in ways consistent with the names of the features. For example, the aggressive feature is uniformly used to produce aggressive behavior. With the feature to behavior mapping thus fixed, facets of a personality can be determined by the mapping from emotion to features.

The standard fear emotion, for instance, might lead to any of a number of features, such as fright, or flight, or even frozen, depending on the artist's choice of the emotion to feature mapping. Each of these features would cause the previously constructed behaviors to react appropriately. Another example might be an agent where goal failures were seen as learning experiences and used to enable the proud feature. This approach takes advantage of the feature mechanism's role as an abstract interface between emotion and action.

There are several components internal to Hap that may also help model personality. An agent that overestimated the likelihood of goal success or failure might be an optimist or pessimist. One that consistently overrated the importance of goals would tend to have extremes of hope, fear, happiness, and sadness. Making an agent's success tests too easily satisfied would produce sloppiness or incompetence, while making its context conditions too difficult to maintain would produce a kind of perfectionism.

5 Conclusion

We have described Tok, an architecture that integrates mechanisms for reactivity, goals, and emotion. Several mechanisms, including behavioral features, success tests, and context conditions, support the integration. Lyotard, a particular agent, has been built in Tok and exhibits signs of success in integration.

While Tok maintains various kinds of memory, including perceptual memory, a richer learning mechanism is conspicuously absent from the architecture. There are two reasons for this. First, Oz worlds exist for only a few hours, perhaps too short a time for interesting learning to occur. Thus, the integration issues discussed here seem more important for our application. Second, the integration of learning with action is widely studied, and we want to build on this substantial effort rather than compete with it. As a result, we may be failing to see essential constraints that could guide us to a better architecture. To help judge this possibility, one of our colleagues is implementing Lyotard in the Soar architecture.

We are engaged in several efforts to extend Tok. First, Gump and Glinda, our natural language components, are attached to Tok only as independent Lisp modules invokable from Hap rules. It would be best if they were expressed as complex behaviors written directly in Hap. We have increasingly observed similarities in the mechanisms of Hap and Glinda, and are exploring the possibilities of merging them fully.

Second, since the Oz physical world is itself a simulation, it would be conceptually straight-forward to embed a (possibly imprecise) copy inside Tok for use as an envisionment engine. This might allow Tok, for instance, to consider possible re-orderings of steps in behaviors, and to make other decisions based on a modicum of foresight.

Finally, we have built and are continuing to build several real-time, multi-agent, animated, interactive Oz worlds. This is imposing hard timing constraints and genuine parallelism on Tok, and has caused substantial changes to the implementation and smaller changes to the architecture.

It has been suggested to us that it may be impossible to build broad, shallow agents. Perhaps breadth can only arise when each component is itself modeled sufficiently deeply. In contrast to the case with broad, deep agents (such as people), we have no *a priori* proof of the existence of broad, shallow agents. However, at least in the Oz domain, where sustained suspension of disbelief is the criteria for success, we suspect that broad, shallow agents may be possible. Tok is an experimental effort to judge the issue.

6 Acknowledgments

This research was supported in part by Fujitsu Laboratories, Ltd. We thank Phoebe Sengers, Peter Weyhrauch, and Mark Kantrowitz for their assistance, and the reviewers for helping us clarify our presentation.

References

[1] Philip E. Agre and David Chapman. What are plans for? In *Robotics and Autonomous Systems*. Elsevier Science Publishers, 1990.

[2] Joseph Bates. Computational drama in Oz. In *Working Notes of the AAAI-90 Workshop on Interactive Fiction and Synthetic Realities*, Boston, MA, July 1990.

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

[4] Joseph Bates, A. Bryan Loyall, and W. Scott Reilly. Broad agents. In *Proceedings of AAAI Spring Symposium on Integrated Intelligent Architectures*, Stanford, CA, March 1991. Available in *SIGART Bulletin*, Volume 2, Number 4, August 1991, pp. 38-40.

[5] Joseph Bates, A. Bryan Loyall, and W. Scott Reilly. 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, July 1992.

[6] Rodney Brooks. Intelligence without representation. In *Proceedings of the Workshop on the Foundations of Artificial Intelligence*, June 1987.

[7] Jaime Carbonell. Computer models of human personality traits. Technical Report CMU-CS-79-154, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, November 1979.

[8] Michael Dyer. *In-Depth Understanding.* The MIT Press, Cambridge, MA, 1983.

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

[10] Michael P. Georgeff, Amy L. Lansky, and Marcel J. Schoppers. Reasoning and planning in dynamic domains: An experiment with a mobile robot. Technical Report 380, Artificial Intelligence Center, SRI International, Menlo Park, CA, 1987.

[11] Eduard Hovy. *Generating Natural Language under Pragmatic Constraints.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1988.

[12] Mark Kantrowitz. Glinda: Natural language text generation in the Oz interactive fiction project. Technical Report CMU-CS-90-158, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1990.

[13] Mark Kantrowitz and Joseph Bates. Integrated natural language generation systems. In R. Dale, E. Hovy, D. Rosner, and O. Stock, editors, *Aspects of Automated Natural Language Generation*, volume 587 of *Lecture Notes in Artificial Intelligence*, pages 13–28. Springer-Verlag, 1992. (This is the Proceedings of the Sixth International Workshop on Natural Language Generation, Trento, Italy, April 1992.).

[14] John Laird, editor. *Proceedings of AAAI Spring Symposium on Integrated Intelligent Architectures*, March 1991. Available in *SIGART Bulletin*, Volume 2, Number 4, August 1991.

[15] John E. Laird and Paul S. Rosenbloom. Integrating planning, execution, and learning in soar for external environments. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 1022–1029. AAAI Press, 1990.

[16] Michael Lebowitz. Story-telling as planning and learning. *Poetics*, 14:483–502, 1985.

[17] A. Bryan Loyall and Joseph Bates. Hap: A reactive, adaptive architecture for agents. Technical Report CMU-CS-91-147, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, June 1991.

[18] James Meehan. *The Metanovel: Writing Stories by Computer.* PhD thesis, Computer Science Department, Yale University, 1976. Research Report #74.

[19] Tom M. Mitchell. Becoming increasingly reactive. In *Proceedings of the Eighth National Converence on Artificial Intelligence*, pages 1051 – 1058. AAAI Press, 1990.

[20] Tom M. Mitchell. Plan-then-compile architectures. In *Proceedings of AAAI Spring Symposium on Integrated Intelligent Architectures*, Stanford, CA, March 1991. Available in *SIGART Bulletin*, Volume 2, Number 4, August 1991, pp. 136-139.

[21] Erik T. Mueller. *Daydreaming in Humans and Machines.* Ablex Publishing Corporation, 1990.

[22] Allen Newell. *Unified Theories of Cognition.* Harvard University Press, Cambridge, MA, 1990.

[23] A. Ortony, G. Clore, and A. Collins. *The Cognitive Structure of Emotions.* Cambridge University Press, 1988.

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

[25] S. Vere and T. Bickmore. A basic agent. *Computational Intelligence*, 6:41–60, 1990.