Representing and Learning Routine Activities

by

Henry H. Hexmoor

December 1995

A dissertation submitted to
the Faculty of the Graduate School of
State University of New York at Buffalo
in partial fulfillment of the requirements for
the degree of
Doctor of Philosophy

Acknowledgments

This work was supported in part by Equipment Grant No. EDUD-US-932022 from SUN Microsystems Computer Corporation, and in part by NASA under contract NAS 9-19004, which is gratefully acknowledged.

I want to thank the members of my committee for their support and patience. My advisor, Dr. Stuart C. Shapiro, gave me the freedom to explore many ideas. He has been very insightful and given me advice, some of which I will appreciate in my future work. Dr. William J. Rapaport made me pin down a lot of the terms and concepts I used loosely. Bill has also tirelessly read most of my writings and suggested improvements in writing and presentation. I will miss his advice. Dr. Bharat Jayaraman put up with reading through my rough drafts. Peter Bonasso Of NASA JSC read my dissertation and made several comments that clarified and improved my dissertation. I thank Johan Lammens for his collaboration in our robotics work. Joe was my officemate and house mate for over two years, and his friendship made Buffalo winters almost bearable. I thank support of my officemate Alistair Campbell. I thank Chris Crowner and his wife Mary-Ann Bliznik for renewing my lost trust in friendship.

I thank the secretaries Ellie Benzel, Sally Elder, and Gloria Koontz for always getting me through various paperwork. I would like to thank the Department of Computer Science for teaching me computer science. I would like to think that my research seamlessly embraces all sciences but that it has benefited by computer science and cognitive science.

I thank Cesar Bandera of Amherst Systems Inc. for his initial interest in my work and collaborations that led to funding my studentship and later to funding that will cover my postdoctoral residence. I hope we will continue to collaborate and enjoy the synergy in our joint work.

I thank Dr. Wayne Book, one of my Engineering mentors at Georgia Tech, for teaching me not to jump in the ocean of research without a life raft. Dr. George Vachsevanos, my other Engineering mentor at Georgia Tech, taught me the ways to innovate and overcome obstacles and that education is all about building a personal tool box. I thank Don Nute, my mentor at the University of Georgia, for believing in my ability more than I did and for teaching me logic. I thank Dr. Barry Silverman of George Washington University for trusting my leadership in research projects and opening his home to me on the occasions

I visited Washington, DC. I thank William Underwood of AI Atlanta for teaching me how to write and conduct research grants.

I thank Nancy Hexmoor for being an example of hard work, unconditional kindness, and dedication. I thank Chelsea Brinkley for finding me when I was lost and lonely. Chelsea and her two boys brought me a whole other world to worry about besides school.

Abstract

A routine is a habitually repeated performance of some actions. Agents use routines to guide their everyday activities and to enrich their abstract concepts about acts.

This dissertation addresses the question of how an agent who is engaged in ordinary, routine activities changes its behavior over time, how the agent's internal representations about the world is affected by its interactions, and what is a good agent architecture for learning routine interactions with the world. In it, I develop a theory that proposes several key processes: (1) automaticity, (2) habituation and skill refinement, (3) abstraction-by-chunking, and (4) discovery of new knowledge chunks.

The process of automaticity caches the agent's knowledge about actions into a flat stimulus-response data structure that eliminates knowledge of action consequences. The stimulus-response data structure produces a response to environmental stimuli in constant time.

The process of habituation and skill refinement uses environmental clues as rewards. Rewards are used to develop a bias for the agent's actions in competing action-situation pairs where the agent did not have prior choices.

The process of abstraction-by-chunking monitors the agent's increased reliance on stimulus-response data structures and turns the agent's complex actions into primitive acts, thereby eliminating the need for the agent to elaborate its plans beyond a certain level.

The process of discovering knowledge-chunks monitors the agent's use of stimulus-action data structures and constructs knowledge about action preconditions and consequences and constructs patterns of interactions into plans.

I have implemented several agents that demonstrate parts of my theory using an agent architecture I developed called GLAIR. GLAIR models agents that function in the world. Beyond my theory about routines, GLAIR is used to demonstrate situated actions as well as deliberative actions. Each of GLAIR's three levels (Sensori-actuator, Perceptuo-motor, and Knowledge) uses different representations and models different components of intelligent agency. GLAIR levels operate semi-autonomously. Whereas intra-level learning improves an agent's performance, inter-level learning migrates know-how from one level to another. Using the GLAIR architecture, I model agents that engage in routines, use these routines

to guide their behavior, and learn more concepts having to do with acts.

The significance of my theory, architectures, and the agents are to illustrate that computational agents can autonomously learn know-how from their own routine interactions in the world as well as learn to improve their performance.

Contents

Chapter 1

Introduction

This dissertation presents a theory of how agents learn in the context of routines. My theory is also useful in building agents with these learning capabilities. I propose an architecture that facilitates study of routine activities. This architecture is used for modeling the interaction of an agent in dynamic environments. I demonstrate my theory with implemented agents.

"*Routines* are patterns of interaction between an agent and its world" [Agre and Chapman, 1987, p. 269]. Getting in or out of a car, changing lanes, and flipping pages of a book can be routines for an agent if the agent consistently engages in these activities in a similar way. In other words, a task for an agent is a routine if the agent who has choices about how to accomplish that task nevertheless does it in the same way or periodically like every day, week, or year. Consistently putting on the left leg of a pair of pants before putting on the right leg would be a routine for an agent.[1]

We see instances of routines as by-products of research in planning, artificial life, and architectures for intelligent agency. Routines have not been direct objects of study. Colloquial use of the term "routine" dominates reference to this word in the relevant literature. I try to understand and model learning about acts, actions, and skills in the context of routines.

Regardless of the benefits of routines, they are either "conscious" or "unconscious." A

[1] I assume the agent has no medical or other conditions that make it choose one leg over another.

“conscious” routine is a set of willful policies, a plan, or a script for doing something. An “unconscious” routine is a sequence of actions produced unconsciously that occurs repeatedly under the same situations. These “unconscious” actions are not involuntary actions like the heart beating. The same routine can be either “conscious” or “unconscious.”

In [Agre, 1988], a *routine* is described as a frequently repeated pattern of activity between an agent and the world, e.g., picking up a fork, making a bowl of cereal, putting on a heavy backpack, selecting a toll booth at the San Francisco Bay Bridge, putting a watch on, breaking an egg into a bowl, washing dishes in the kitchen sink after a large dinner, tossing a wad of paper into the trash can in the far corner of the office, and writing the word “the”. Agre points out that he might not have any fixed way of pouring liquids, but in his morning stupor he might have a set way of pouring his first cup of coffee. He goes on to emphasize the improvisational aspects of routines. However, he never gets around to suggesting that we learn from these routines. I believe that routines consist of a set of preferences adhered to habitually. Agre does not examine the evolution of a routine from what I believe to be its constituent habitual preferences. Habitual preferences are choices about non-consequential preferences, like picking a lane at the toll booth given that all lanes are equal in all toll-paying respects. A course of action or a set of tendencies for choosing among alternative actions can be a routine that is useful in many occasions and is part of many tasks. On the one hand, such a course of action can be considered to be a skill.[2] Agre does not look at mechanisms for identifying and using skills by the agent. I sketch learning techniques that address learning skills from routine activities.

Since I am concerned with everyday activities, I consider situated actions [Suchman, 1988]. A characteristic of these actions is that an agent exhibits an action without reasoning or making “conscious” decisions; therefore, situated actions are more like “unconscious” behaviors than deliberated actions. The following quote illuminates that a situated action is a “real time” tendency to act, which is a feature of “unconscious” behaviors: “The coherence of situated action is tied in essential ways not to individual predispositions or conventional rules but to local interactions contingent on the actor’s particular circumstances” [Suchman, 1988, p. 27]. On the other hand, a routine might be a plan, a set of policies, or a script to be followed. I consider plans and scripts to be routines at the “conscious” level.

[2] In computational terms, this is like a macro.

In memory-based planning [Alterman, 1986, Hammond, 1986], old plans (cases) are modified to suit new situations, and from this new plans are learned. The problem in memory-based planning is debugging the ordering of acts in a plan. Often a few of the steps in a plan may need to be excluded, reordered, or modified. Our problem is not about modifying plans. I am interested in working at the level of acts and how they can be adapted and learned.

Following the research in situated activity [Agre and Chapman, 1987, Suchman, 1988, Nilsson, 1992], in dynamic domains, where reactions are needed in less time than it takes to plan an action, an agent operates with situated actions instead of planned actions. In the larger context of intelligent agency, I argue that situated activity is a particular form of interaction between an agent and its environment that is produced by an agent when it is involved in "unconscious" interaction [Hexmoor et al., 1992, Hexmoor et al., 1993a, Hexmoor et al., 1993b]. Elsewhere [Hexmoor and Shapiro, 1995], we discuss how "unconscious" actions are part of an agent's skills as opposed to its knowledge. "Unconscious" routines can be considered to be skills. On the other hand, an agent may also engage in "conscious," reasoned-out interaction. In collaboration with Stuart C. Shapiro and Johan Lammens, I have developed an architecture that facilitates modeling "conscious" and "unconscious" interactions of an agent with the world. I will discuss details of my architecture in chapter 4. For now I will merely point out that this architecture, GLAIR, has three levels: the Knowledge level, the Perceptuo-Motor level, and the Sensory-Actuator level. The latter two levels model "unconscious" interactions, whereas the first level models "conscious" interactions.

I digress at this point to talk about my distinction between knowledge and skills and how in my use of those terms, knowledge falls in the "conscious" realm whereas skills are considered to be "unconscious." I consider a subset of an agent's mental/physical abilities represented by a strong symbol system which I call the agent's knowledge. I consider the remaining abilities to be skills. This definition make our use of those terms technical and different from colloquial use. For example, brushing teeth in the morning is mostly an "unconscious" activity. However, this task is not considered to be a skill. In my methodology, this would be a skill and an "unconscious" activity. The colloquial use of the term skills implies that: (1) a skill is a task which is learned by practice, (2) there is a barometer of success, and (3) it is a social privilege to have a skill. Many real world example have both components of skill and knowledge and have “conscious” and “unconscious” elements. An example is cooking. Cooking entails knowledge of recipes and directions which is “conscious.” Cooking has a “unconscious” component of interacting with kitchen-ware and food. I will consider the dynamic aspects of interaction in cooking to be skills.

In light of my distinction of “conscious” and “unconscious” interaction, I will specify the impact of routines on “conscious” and “unconscious” interactions with the environment. In general, I am interested in ways in which an agent discovers, learns, and uses routines to make it more reactive. In the following sections, I will outline discovering, learning, and using routines. I will defer details of how routines are represented and used until I have introduced an architecture for intelligent agency in a later chapter.

1.1 Definition

The following is my definition of a routine:

A routine is a course of action that is habitually (or by choice) adhered to.

The examples I cited in the beginning of this chapter from [Agre, 1988] are ones I would use for this definition. By adherence to a course of action I mean that the agent reliably selects the same course of action in similar circumstances. This definition assumes an agent that lives in a world and performs a set of tasks that are part of its life. I assume the agent starts out with no a priori plan or preference for a course of action. By engaging in tasks, the agent develops preferences for actions. I envision a natural way for developing a routine to be a new task. If the agent has determined preferences for all possible choice points in the space of possible actions, then the agent has a routine for the task. The world of the agent may or may not be static or changing. If an agent develops a routine in a particular world and then the world changes, the routine may be ineffective. As an example of a routine that can be ineffective, consider an agent who develops a routine for finding an object in a cluttered but static room; the agent basically has learned the layout of the room. If we now move objects in the room, the agent needs to develop or extend other routines for finding an object. However, I want the new routine to work in the previous setting as well. In other words, I would like routines to endow the agent with ever more preferences. Similarly, if the world becomes dynamic, e.g., the agent interacts with moving objects, new routines should cope with previously static situations. However, I don't expect the routines to be so robust that I can change the agent's morphology, e.g., change its sensors and actuators, and expect the new routines to cope with the changed morphology.

Since a routine is a course of action or a set of preferences, within this course of action, there will be subsets of courses of action that may remain the same even across morphology changes. For example, an agent that maneuvers around corners in a smoothed arc motion instead of a stop-turn-start sequence should be able to keep this ability over many task spaces and many minor morphological changes.

Many recent works in mobile robotics have addressed learning isolated actions, particularly through neural network approaches, e.g., [Nehmzow, 1992]. In my approach, I don't isolate actions for learning. Instead I focus on emergent sequences of actions that are parts of larger tasks and unsupervised learning of these action sequences. Furthermore, sequences of actions learned by the agent become significant for the agent after being learned, and the agent generates concepts corresponding to these sequences to be used in subsequent interactions or reasoning.

1.2 Uses of Routines

There are times that we notice our habitual behaviors, like tying a shoelace, in a particular way. Why or when we form habits are beyond the scope of my interest. Habits are "unconscious" preferences. Habits, like routines, can either be "conscious" or "unconscious." I consider habits to be a kind of routine. A nonhabitual routine is like a procedure like the procedure for paying the fine on a library book. Here I seek ways to use and at times discover routines that are habitual.

I expect routines to be used in one of three ways: (a) as mental notions to be reasoned about, (b) as stand-alone "unconscious" pieces of behavior, (c) as tendencies for choosing certain behaviors over other applicable behaviors.

1.2.1 Routines Are Used to Guide Daily Activity

A routine is either imposed upon the agent (a plan at the "conscious" level to be followed), in which case it need not be discovered, or it is performed by the agent automatically, i.e., unconsciously. If it is performed unconsciously, it may or may not ever be discovered, i.e., noticed and made "conscious" However, the existence of such a routine may guide the agent's actions. If it remains "unconscious," it aids in choosing among competing actions unconsciously as an unexplained tendency or a preference. If it is noticed and made "conscious," it can also be used as a concept and used in reasoning about actions and planning.

An important feature of GLAIR is that it allows knowledge to migrate from one level to another and, in doing so, the knowledge changes representation in order to be consistent with the representation used at that level. The underlying assumption is that parts of an agent's knowledge about its world have natural loci either in the "conscious" or the "unconscious." As the agent interacts with the world, the knowledge may be gained at some level and later find its natural locus at a different level and migrate there. The migration of knowledge from the "conscious" level to the "unconscious" level often is a result of caching (i.e, storing for future use) reasoning that takes place at the "conscious" level. Thus, what is migrated to the "unconscious" is cached reasoning or just chunks of "what to do when". The result of this migration is that the agent stops reasoning earlier and uses a chunk which expands at the "unconscious." *Chunk* is a psychological term. Chunking allows the agent quickly to pick an action without deliberation. This lack of deliberation is a form of lessening the cognitive burden of deciding "what to do next". Naturally this lessening cognition aids in guiding the agent's actions.

I am interested in all routines, including ones that are not discovered but adopted. Reading a driver's manual, we are told about passing another vehicle, and although that may not be the only way to pass another vehicle we adopt it. Once our "conscious" knowledge of passing another vehicle is practiced, it becomes part of our skills; in the framework of GLAIR I consider it to have become "unconscious." I can consider this process of learning as a form of caching a plan into reactive plans as was done in [Schoppers, 1987, Drummond, 1989]. I will review and compare these techniques in the section on related work.

Another way that the agent's actions are guided by routines is by refining skills at the "unconscious" level. There are many routines that need not have a concept associated with them at the "conscious" level, but nevertheless are useful in guiding actions. These are tendencies that the agent may fall into and stay with. Informally, I consider this use of routines as skill acquisition.

1.2.2 Routines Are Used to Enrich an Agent's Knowledge and Skill with Acts

Beyond their capacity to guide an agent's activity, routines can enrich an agent's ability to interact with the world. This enrichment takes two forms. The first is to enrich the agent's internal model of its acts. This means that the agent learns more about its abilities and represents these abilities in its repertoire of acts. The second method is an enrichment that does not affect the agent's explicit model of the world. In other words, no modifications are made to what is represented. Instead, the agent improves its ability in choosing among alternate actions. With both methods, the agents learns more complex acts and behaviors.

1.3 The Problem

It is not practical to design a general-purpose agent that already knows how to interact effectively in all possible environments. Building such an agent is like building an adult person who has had lots of experience interacting in unpredictable environments. Instead, I focus on building agents with initial primitive capabilities that the agent can adapt and acquire other capabilities through interaction with the environment.

In *domain-independent planning* [Wilkins, 1984], the domain knowledge is excluded from planning knowledge. I believe that it is severely limiting to provide an agent with domain-independent planning, by giving it a planner, without also giving it an ability to reason about its planning, abstracting, and augmenting its planning knowledge. I advocate building an agent with the ability to reason about planning and to extend its planning through interaction with the environment.

Routines are products of interactions between an agent and its environment. As such, the agent can only have an impression of the interaction and not full knowledge of how it and the environment evolve in their interaction. I am interested in a theory that explains how an agent is influenced by the routines. This influence can either impact the agent's knowledge and its skills. In addition to explanatory capacity of this theory, I require this theory to be used as a model of developing agents that exhibit improved interactions and enriched internal representations. This theory should answer the following questions:

- What are the impacts of routines on an agent's internal states and representations? Architecturally, how can such an agent be constructed?
- How does an agent improve its interactions in the world? Architecturally, how can such an agent be constructed?
- What is a good agent architecture for exploring learning routine interactions in the world?

How the agent is affected by learning routines depends a lot on the internal data structures used in modeling the agent and the organization of cognitive and sub-cognitive components of the agent prior to learning. I use Herbert Simon's definition of learning, in which Simon says that learning is a change in the agent's internal structure in a way that affects the agent's long-term performance. A superior data structure and organization will have a clear locus of information transfer and changes in the agent. Issues about data structure and organization are asked in my above question about fitness of an agent architecture for learning routines.

1.4 Plan of the Dissertation

I will define the problems to be addressed in section 1.3 of this chapter. The remainder of my dissertation has two major parts. The first part contains chapters 2, 3, and 4 in which I present a specific architecture and a theory of routines. In chapter 2, I introduce my architecture within which I demonstrate routines. My architecture provides a framework for considering the impact of routines on "conscious" and "unconscious" processes. My philosophy of intelligent agency presents my position on the hotly debated issue of situated cognition as well as on behavior-based AI. Chapter 2 forms an important background for understanding my approach to the study of routines. At the end of chapter 2, I review related work on architectures. The discussion of issues related to my architecture took place in a AAAI symposium Dr. David Kortenkamp and I organized in March 1995 at Stanford, CA. I omit discussions of lessons I learned from implemented agents as well as open problems I discovered. The interested reader can see [Hexmoor, 1995b] or [Hexmoor and Kortenkamp, 1995]. I review background related to my theory of routines in chapter 3. I then present a theory of impact of routines on the mind in chapter 4. This theory synthesizes four processes that contribute to various forms of learning in the context of routines.

The second part of this dissertation is in chapters 5 and 6 in which I selectively present implementations of agents and discussions of how those agents address the theory of routine. I omit several implementations of GLAIR-based agents. The interested reader can see [Hexmoor et al., 1993b]. In chapter 5, I describe implementations of agents that conform to my architecture. These agents are cast in environments that engage them in routines. Chapter 6 present experiments in implementing GLAIR and the four processes in my theory described in chapter 4. I give examples of these processes using implemented GLAIR agents. My implementations demonstrate a few salient features of my theory presented in chapter 4. I will point out the parts that are not yet implemented.

Chapter 7 concludes what is achieved by my theory of routines and my architecture. Chapter 8 outlines the loose ends and extensions of this work as well as implications for related fields.

Chapter 2

GLAIR: An Architecture for Autonomous Agents

2.1 Introduction and Overview

I developed GLAIR (Grounded Layered Architecture with Integrated Reasoning) in collaboration with Stuart C. Shapiro and Johan Lammens. This chapter is a summary of [Hexmoor et al., 1993a, Hexmoor et al., 1993b, Hexmoor et al., 1993c, Hexmoor et al., 1993d, Hexmoor, 1994, Hexmoor and Shapiro, 1995, Lammens et al., 1995, Hexmoor, 1995a, Hexmoor and Kortenkamp, 1995]. Johan Lammens contributed most to the ideas on symbol grounding and embodied representation.

By an *architecture*, I mean an organization of components of a system, what is integral to the system, and how the various components interact.[1] Which components go into an architecture for an autonomous agent has traditionally depended to a large extent on whether we are *building a physical system*, *understanding and modeling the behaviors of an anthropomorphic agent*, or *integrating a select number of intelligent behaviors*. The organization of an architecture is also influenced by adopting the *modularity* assumption of Fodor [Fodor, 1983], or a *connectionist* point of view, e.g., [McClelland et al., 1986], or an *anti-modularity*

[1]Our discussion of architecture in this chapter extends beyond any particular physical or software implementation.

assumption as in Brooks's subsumption architecture [Brooks, 1985]. The *modularity* assumption supports (among other things) a division of the mind into a *central system*, i.e., cognitive processes such as learning, planning, and reasoning, and a *peripheral system*, i.e., sensory and motor processing [Fodor, 1983]. Our architecture is characterized by a three-level organization into a Knowledge Level, a Perceptuo-Motor Level, and a Sensory-Actuator Level. This organization is neither modular, anti-modular, hierarchical, anti-hierarchical, nor connectionist in the conventional sense. It integrates a traditional symbol system with a physically grounded system, a *behavior-based* architecture. The most important difference from a behavior-based architecture like Brooks's subsumption architecture is the presence of three distinct levels with different representations and implementation mechanisms for each, particularly the presence of an explicit Knowledge Level. Representation, reasoning (including planning), perception, and generation of behavior are distributed through all three levels.

Architectures for building physical systems, e.g., robotic architectures [Albus et al., 1981], tend to address the relationship between a physical entity (e.g., a robot), sensors, effectors, and tasks to be accomplished. Since these physical systems are performance centered, they often lack general knowledge representation and reasoning techniques. These architectures tend to be primarily concerned with the *body*, that is, how to get the physical system to exhibit intelligent behavior through its physical activity. These systems are not concerned with *"consciousness"*. These architectures address what John Pollock calls *Quick and Inflexible* processes [Pollock, 1989]. Operationally, a robotics agent is "conscious" if it is aware of its environment, as evidenced by (1) having some internal states or representations that are causally connected to the environment through perception, (2) being able to reason explicitly about the environment, (3) being able to communicate with an external agent about the environment, and (4) having internal states of a sense of self. Simply being able to verbalize a thought is not a defining criteria.[Shepard and Metzler, 1971]. Also, having a response for stimulus is not adequate for consciousness [Culbertson, 1963]. A vending machine or an industrial robot has responses, but it is not conscious of the interaction.

Architectures for understanding and modeling behaviors of an anthropomorphic agent, e.g., cognitive architectures [Anderson, 1993, Pollock, 1989, Langley et al., 1991], tend to address the relationships that exist among memory, reasoning abilities, intelligent behavior, and mental states and experiences. These architectures often do not take the *body* into account. Instead they primarily focus on the *mind* and *"consciousness"*. Our architecture ranges from general knowledge representation and reasoning to body-dependent physical behavior.

I am interested in autonomous agents that are embedded[2] in a dynamic environment. Such an agent needs to interact continually with and react to its environment and exhibit intelligent behavior through its physical activity. To be successful, the agent needs to reason about events and actions in the abstract as well as in concrete terms. This means combining situated activity with acts based on reasoning about goal-accomplishment, i.e., deliberative acting or planning. In the latter part of this dissertation, I will present a family of agents based on my architecture. These agents are designed with a robot in mind, and their structure is also akin to anthropomorphic agents. Figure 2.1 schematically presents our architecture.

I distinguish between deliberative, reactive, and reflexive behaviors. Embodied representations at the Perceptuo-Motor Level facilitate this integration. As we move down the levels, computational and representational power is traded for better response time and simplicity of control. The agent learns from its interactions with the environment. It has a capacity for *engaged* and *disengaged reasoning*. The former occurs when behavior is generated directly while reasoning, in a lock-step fashion. The latter occurs when reasoning is done in a hypothetical mode, not directly generating behavior.[3]

There are several features that contribute to the robustness of my architecture. I highlight them below (an in-depth discussion follows later):

- I differentiate "conscious" reasoning from "unconscious" Perceptuo-Motor and Sensori-Actuator processing. I consider body-related processes to be *"unconscious"*, but that is not meant to imply anything about their complexity or importance to the architecture as a whole. Indeed, I believe that the "unconscious" levels of our architecture (the Perceptuo-Motor level and the Sensori-Actuator level) are at least as important to the architecture as the "conscious" one (the Knowledge level). I reserve the

[2] "Embedded agents are computer systems that sense and act on their environment, monitoring complex dynamic conditions and affecting the environment in goal-oriented ways." [Kaelbling and Rosenschein, 1990, p. 1].

[3] Elsewhere, I discuss alignment as a mechanism that allows me to elegantly model both modes of reasoning [Hexmoor et al., 1992].

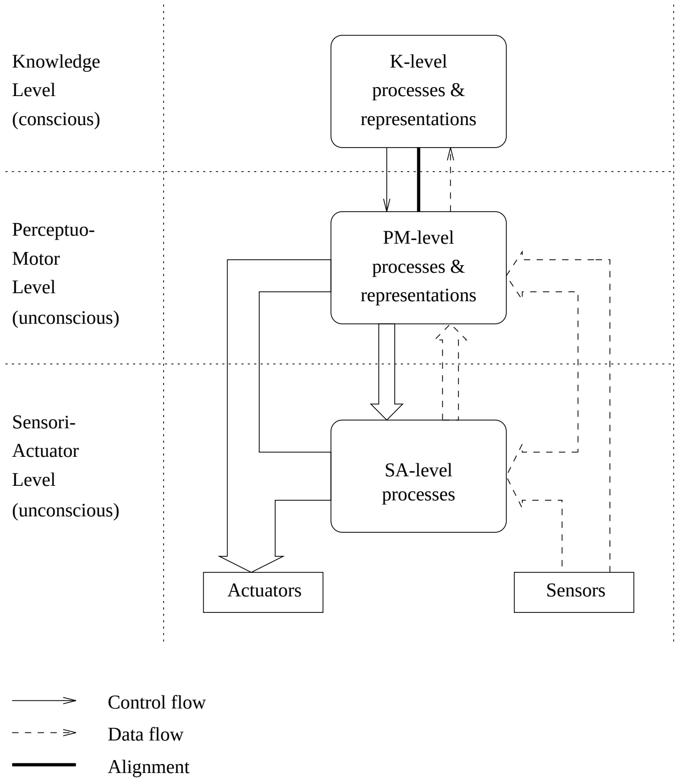

Figure 2.1: Schematic representation of the agent architecture. Width of control and data paths suggests the amount of information passing through (bandwidth). Sensors include both world-sensors (sensors that gather information from the world) and proprio-sensing (sensors that gather information from the agent's internal organs) sensors.

term *sub-conscious* for implicit cognitive processes such as category subsumption in Knowledge Representation and Reasoning (KRR) systems. See [Shapiro, 1990] for a discussion of sub-conscious reasoning.

- The levels of our architecture are semi-autonomous and processed in parallel. This autonomy is similar to Brooks's subsumption architecture [Brooks, 1985], but at a more macroscopic level. Brooks does not distinguish between the three levels I describe, since his work is solely concerned with behaviors whose controlling mechanism I would situate at the Perceptuo-Motor level.

- "Conscious" reasoning takes place through explicit knowledge representation and reasoning. "Unconscious" behavior makes use of several different mechanisms.

- "Conscious" reasoning guides the "unconscious" behavior, and the "unconscious" levels, which are constantly engaged in perceptual and motor processing, can *alarm* the "conscious" level of important events, taking control if necessary. Control and generation of behavior are decomposed into levels and not exclusively top-down.

- Lower-level mechanisms can pre-empt higher-level ones. This appears to turn subsumption (when activated, the higher behaviors taking over lower behaviors) on its head, but everything depends on the placement of behaviors in the hierarchy, of course. I haven't quite decided yet whether inhibition should work the other way around as well.

- There is a correspondence between concepts in the Knowledge Representation and Reasoning (KRR) system, on the one hand, and sensory perceived objects, properties, events, and states of affairs in the world and motor capabilities, on the other hand. I call this correspondence *alignment*.

In [Brooks, 1990], Brooks criticizes the ungroundedness of traditional symbolic AI systems, and proposes physically grounded systems as an alternative, particularly the subsumption architecture. Subsumption has been highly successful in generating a variety of interesting and seemingly intelligent behaviors in a variety of mobile robots. As such, it has established itself as an influential approach to generating complex physical behavior in autonomous agents. In this dissertation, I explore the possibilities of integrating the old with the new in an autonomous-agent architecture that ranges from physical behavior generation inspired by subsumption to classical knowledge representation and reasoning, and a new proposed level in between the two. Although I am still struggling with many of the issues involved, I believe I can contribute to a solution to some of the problems for both classical systems and the physically grounded systems mentioned in [Brooks, 1990], in particular:

- The ungroundedness of symbolic systems (referred to as "the symbol grounding problem" in [Harnad, 1990]): our architecture attempts to ground high-level symbols in perception and action through a process of embodiment.

- The potential mismatch between symbolic representations and the agent's sensors and actuators: the embodied semantics of our symbols makes sure that this match exists.

- Our symbolic representations do not have to be named entities. The knowledge representation and reasoning system I use in my implementations allows the use of unnamed intensional concepts.
- I have some ideas about how to automate the construction of behavior-generating modules through learning, but much remains to be done.

I agree with the requirement of physically implemented systems as the true test for any autonomous agent architecture, and to this end I am working on several different implementations. I will present both our general multi-level architecture for intelligent autonomous agents with integrated sensory and motor capabilities, GLAIR, and a physical implementation and two simulation studies of GLAIR-agents.

2.2 The GLAIR Architecture

In this section I discuss in detail our autonomous agent architecture for integrating perception and acting with grounded, embodied, symbolic reasoning.

2.2.1 Architecture Levels

I now proceed to discuss one of the distinguishing characteristics of GLAIR: its three levels.

2.2.1.1 Motivation

The three levels of our architecture are of organizational as well as theoretical importance. The layered architecture allows us to work on individual levels in a relatively independent manner, although all levels are constrained by the nature of their interactions with the adjoining level(s). The architecture is hierarchical, in that level i can only communicate with levels $i-1$ and $i+1$, if any.

The levels of our architecture are semi-independent. While control flows mainly top-down and data mainly bottom-up, local control mechanisms at any level can preempt higher-level control, and these local mechanisms filter the data stream for their own purpose, in parallel with higher-level ones. Representations become coarser-grained from bottom to top, while control data becomes more fine-grained from top to bottom. The terms in the

Knowledge Level's KRR system model "conscious" awareness of the world (and the body), and the perception and motor capabilities in the other levels provide the grounding for an *embodied semantics* of the terms in the knowledge level. Ordinary, reflex-like activities are controlled by close coupling of perception with motor actions at the ("unconscious") Perceptuo-Motor and Sensori-Actuator levels. This close coupling avoids having to exert control over these activities from the "conscious" level, as in purely top-down structured architectures with a symbol level at the top of the hierarchy. In the latter kind of system, signals must first be transformed to symbols and vice versa. The low-level coupling provides for better real-time performance capabilities, and relieves the Knowledge level of unnecessary work.

In general, GLAIR has multi-level layered representations of objects, properties, events, states of affairs, and motor capabilities, and the various levels are *aligned*. By alignment, we mean a correspondence between representations of an entity at different levels. This organization contributes to the robustness and computational efficiency of implementations. The semi-autonomous nature of the levels allows for graceful degradation of system performance in case of component failure or situation-dependent failure. Lower levels can function to some extent without higher-level control, and higher levels can function to some extent without lower-level input.[4]

Our architecture allows us to elegantly model a wide range of behaviors: from mindless, spontaneous, reflex-like, and automatic behavior, e.g., "stop if you hit an obstacle," to plan-following, rational, incremental, and monitored behavior, e.g., "Get in the car now, if you want to go to LA on Friday."[5]

In anthropomorphic terms, we identify the Knowledge level with consciously accessible data and processing; the Perceptuo-Motor level with "hardwired", not consciously accessible processing and data involved with motor control and perceptual processing; and the Sensori-Actuator level with the lowest-level muscular and sensor control, also not consciously accessible. The substrate of grounding and embodiment [Lakoff, 1987, Suchman, 1988, Har-

[4] For instance, in the context of autonomous vehicles, if obstacle avoidance or returning to the base is a lower-level behavior than planning exploration strategies, then a failure of the hardware implementing the latter does not necessarily prevent the former.

[5] The plan is to get in the car to go to the travel agency to get a ticket to fly to LA on Friday. Today is Thursday and it is near the end of the business day. Also, the agency won't accept telephone reservations. This example is suggested in [Pollock, 1992].

nad, 1990] of actions, concepts, and reasoning is mainly the Perceptuo-Motor level and to some extent the Sensori-Actuator level.

I will now explore representation and computation at the individual levels in more detail.

2.2.1.2 The Knowledge Level

My use of the *Knowledge Level* is similar to Newell's [Newell, 1981] in the sense that we are considering abstract concepts at this level. Whereas Newell's Knowledge Level is a description and does not include implementation, our Knowledge Level is fully implemented, contains a traditional KRR and/or planning system, using a relatively course-grained representation of objects, events (including actions), and states of affairs. For instance, objects are represented at the Knowledge Level as unique identifiers, typically without further detail about their physical characteristics or precise locations. It is possible to represent such detail explicitly at this level, but it is not required. Only if the detail becomes important will it be represented, though not necessarily in the same way as at a lower level. For example, knowledge about the physical size and weight of an object might become available at the Knowledge Level through the agent's actively using measuring devices like a ruler or a scale, but this knowledge is not the same as the embodied knowledge about dimensions and weight represented at the Perceptuo-Motor Level for the particular object or its object class. As a rule of thumb, representations at the Knowledge Level are limited to objects, events, and states of affairs that the agent needs to be consciously aware of in order to reason and plan and in order to communicate with other agents at the grain size of natural language. The Knowledge Level can be implemented using different KRR and/or planning systems.

Traditional use of the concept of world modeling refers to building models of interactions between the agent and its environment at the "conscious" level. These models maintain internal states for the agent. The difference in my use of the term "world model" is that I do not intend to have a precise model of all objects in the environment. Instead, I want to model only the entities relevant to the agent's interaction with its world. This requires filtering out some details accessible at the Perceptuo-motor-level as the entities are aligned with their counterparts on the Knowledge Level. This is known as "perceptual reduction". Physical details of interaction with entities are handled at the Perceptuo-motor level. Representations at the Knowledge level are needed only for explicit reasoning about entities, and contain only the information necessary for doing so. That might include details about physical characteristics in some cases, but it need not. In other cases, it may be limited to a nondescript intensional representation of an object.[6] Conversely, some entities may be represented at the Knowledge Level but not at the Perceptuo-Motor Level (abstract concepts, for instance). Knowledge Level representations are needed for reasoning about entities; Perceptuo-Motor Level representations are needed for physically interacting with entities.

2.2.1.3 The Perceptuo-Motor Level

The *Perceptuo-Motor Level* uses a more fine-grained representation of events, objects, and states of affairs. For instance, such things as size, weight, and location of objects on the kinematic side, and shape, texture, color, distance, pitch, loudness, smell, taste, weight, and tactile features on the perceptual side are specified. At this level, enough detail must be provided to enable the precise control of actuators, and sensors or motor memory must be able to provide some or all of this detail for particular objects and situations. The Perceptuo-Motor Level is partly *aligned* with the Knowledge Level, in that there is a correspondence between object identifiers at the Knowledge Level and objects at the Perceptuo-motor Level.

Kinematic and perceptual representations of particular objects or typical object-class instances may be unified or separate, and both kinds of representations may be incomplete. Also at this level are elementary categorical representations the kinds of representations that function as the grounding for elementary grounded symbols at the Knowledge Level, i.e., sensory-invariant representations constructed from sensory data by the perceptual processor [Harnad, 1990].

The representations at this level are *embodied* (cf. [Lakoff, 1987]), meaning that they depend on the body of the agent, its particular dimensions and characteristics. Robots will therefore have different representations at this level than people would, and different robots will have different representations as well. These representations are agent-centered and agent-specific. For instance, they would not be in terms of grams and meters, but

[6] See [Shapiro and Rapaport, 1986] for our use of "intensional representation".

in terms of how much torque to apply to an object to lift it,[7] or what percentage of the maximum to open the hand to grasp an object. Weights of things in this kind of representation are relative to the agent's lifting capacity, which is effectively the maximum weight representable. An agent may have a "conscious" (Knowledge Level) understanding and representation of weights far exceeding its own lifting capacity, but that is irrelevant to the Perceptuo-Motor Level. When it comes to lifting it, a thousand-pound object is as heavy as a ten-thousand-pound one, if the capacity is only a hundred or so. Similarly, sizes are relative to the agent's own size. Manipulating small things is not the same as manipulating large things, even if they are just scaled versions of each other. A consequence of using embodied representations is that using different "body parts" (actuators or sensors) requires different representations to be programmed or (preferably) learned. While that may be a drawback at first, once the representations are learned they make for faster processing and reactive potential. Representations are direct; there is no need to convert from an object-centered model to agent-centered specifications. This makes the computations at this level more like table lookup than like traditional kinematics computations, which can be quite involved. Learning new representations for new objects is also much simpler; it is almost as easy as trying to grasp or manipulate an object, and merely recording one's efforts in one's own terms. The same holds, *mutatis mutandis*, for perceptual representations.

There are a number of behaviors that originate at this level: some are performed in service of other levels (particularly deliberative behaviors), some are performed in service of other behaviors at this level, a few are ongoing, and some others yet are in direct response to external stimuli. An agent may consciously decide to perform Perceptuo-Motor actions such as looking, as in *look for all red objects*, or to perform a motor action, such as *grasp a cup*. These actions originate at the Knowledge Level and are propagated to this level for realization. An agent has to perform special perceptual tasks to serve other behaviors, such as to *find the grasp point of a cup* in order to *grasp a cup*. These perceptual tasks may originate at this or another level.

---

[7]Of course this also depends on how far the object is removed from the body, or how far the arm is stretched out, but that can be taken into account (also in body-specific terms). People's Perceptuo-Motor Level idea of how heavy something is is most likely not in terms of grams, either (in fact, a "conscious" estimate in grams can be far off), but in terms of how much effort to apply to something to lift it. That estimate can be off, too, which results in either throwing the object up in the air or not being able to lift it at the first attempt, something I have all experienced. On the other hand, having a wrong "conscious" estimate of the weight of an object in grams does not necessarily influence one's manipulation of the object.

At the Perceptuo-Motor Level, an agent has a close coupling between its behaviors, i.e., responses, and stimuli, i.e., significant world states. In nature, we observe that, for a typical agent, there are a finite (manageably small) number of primitive ("innate") behaviors available. As the agent interacts with its environment, it may learn sophisticated ways of combining its behaviors and add these to its repertoire of primitive behaviors. We will consider only an agent's primitive abilities for now. We further assume that the agent starts out with a finite number of ways of connecting world states to behaviors, i.e., reflex/reactive rules. Following these observations, We suggest that at this level, the agent's behavior-generating mechanism is much like a finite state automaton. As we noted earlier, learning will change this automaton. The agent starts with an automaton with limited acuity, and uses its "conscious" level to deal with world states not recognizable at the Perceptuo-Motor Level. For instance, the Perceptuo-Motor Level of a person beginning to learn how to drive is not sophisticated enough to respond to driving conditions automatically. As the agent becomes a better driver, the "conscious" level is freed to attend to other things while driving. This is called *automaticity* in psychology. I discuss an implementation mechanism for these automated behaviors later in this dissertation.

2.2.1.4 The Sensori-Actuator Level

The *Sensori-Actuator Level* is the level of primitive motor and sensory actions, for instance "move from $\langle x, y, z\rangle$ to $\langle x', y', z'\rangle$ or "look at $\langle x, y, z\rangle$". At this level, there are no object representations as there are at the Knowledge Level and the Perceptuo-Motor Level. There are no explicit declarative representations of any kind, only procedural representations (on the actuator side) and sensor data (on the sensory side). Primitive motor actions may typically be implemented in a robot control language like VAL, and some elementary data processing routines may be implemented in a sensory sub-system, like dedicated vision hardware. At this level, we also situate *reflexes*, which we consider to be low-level loops from sensors to actuators, controlled by simple thresholding devices, operating independently of higher-level mechanisms, and able to pre-empt the latter. We see reflexes as primitive mechanisms whose main purpose is prevention of damage to the hardware, or to put it in anthropomorphic terms, survival of the organism. As such, often they take precedence over any other behavior. In humans, "conscious" control can override some reflexes. When reflexes are triggered, the higher levels are made "aware" of this by the propagation of a signal, but they have no control over the reflex's execution, which is brief and simple (like a withdrawal reflex seen in people when they unintentionally stick their hand into a fire).[8] After the completion of a reflex, the higher levels regain control and must decide on how to continue or discontinue the activity that was interrupted by the reflex. Reflex-like processes may also be used to shift the focus of attention of the Knowledge Level.

2.2.2 Deictic (Indexical-Functional) Representations

Objects may be present for the first time in the agent's environment. Objects may be in motion. There may be several objects that are perceptually identical. In general, there may be a large number of objects in the environment of the agent. In these circumstances, it is inappropriate to assign unique names for each object, as is presupposed in traditional model theory. In this section, I describe a class of representations that address the *reference problem* in interacting with a dynamic world. This problem is to find and maintain a correspondence between an agent's representations and objects in its surroundings. In planning, this type of representation is called *indexical-functional* [Agre and Chapman, 1987] or deictic [Agre, 1988], since it relates the referent of a representation to its situation of use (indexicality [Chapman, 1990]), and it relates the referent of a representation to the activity in which the agent is is engaged (functionality).

The advantages of using deictic (indexical-functional) representation over traditional KRR techniques are that:

- In realistic domains, there will be a large number of objects. There is no need to have a representation for each object in the world.
- There is no need for instantiation of object symbols to their extensions in the world.[9]

[8] An appropriate reflex for a robot (arm) might be to withdraw or stop when it meets too much mechanical resistance to its movement, as evidenced for instance by a sharp rise in motor current draw. Such a reflex could supplant the more primitive fuse protection of motors, and make an appropriate response by the system possible. Needless to say, a robot that can detect and correct problems is much more useful than one that merely blows a fuse and stops working altogether. The fact that the withdrawal reflex may not be as strong, or not present at all, when doing this intentionally may point to the need for top-down inhibition as well.

[9] This is a complaint against traditional KRR systems. Our concept of alignment between levels is immune to this objection. Instances in the KL are created only when there is sensory information below the KL that requires instantiation at the KL. GLAIR agents never first generate an instance of a concept and then look for it in the world.

The referent in the world is (by necessity) only indirectly considered via its Perceptuo-Motor Level representation; hence, the problem becomes one of aligning the Knowledge Level representations with the Perceptuo-Motor Level representations. Deictic representation is appropriate for this task. In my treatment of a situated agent (section 3.3.1), I give an account of a KRR system using deictic representations. In the remainder of this section, I will (for the purpose of our discussion) consider physical objects as referents of symbols, keeping in mind that the real referents of Knowledge Level symbols are Perceptuo-Motor Level representations.

Using deictic representation objects are not uniquely identified with symbols. Instead, a deictic symbolic *name* is constructed for each object, as if to answer three questions about it:

1. What type of object is it? E.g., *a coin*, *a cup*, *a saucer*.

2. What is the relation/relevance of the object to the agent? E.g., *in the pocket*, *in front of*.

3. What is the role of the object in the agent's current task? E.g., if the agent is *drinking* from a cup, the cup's role is *drinking-from*; if the agent is *using a fork to lift an object*, the fork is *used-in-lifting-an-object*.

Putting it together, we have descriptions like a-cup-in-front-of-me-that-I-am-drinking-from. Deictic *names* are useful only in so far as they limit the focus for processing and reasoning. A mnemonic form can be generated for human inspection of the functioning of the agent under development.

Let's consider two cases of how indexical-functionals can be generated and used.[10] In the first case, let's consider an object that is in the immediate sensing range; in the second case, we seek an object in the field of view. In the first case, we assume that the agent's sensory-perceptual mechanism automatically identifies the object and creates a representation for it. This is what we refer to as an ongoing behavior in the Perceptuo-Motor Level. Furthermore, we assume that at the Knowledge Level, the first and second questions about the object are trivially answered, in our example, an-apple-on-the-table is generated. Since (as a simplifying assumption) we *index* all objects in the visual field, this eliminates the need to

[10]We have not yet tested the suggestions in this section.

index objects as suggested in [Agre, 1988] in response to actions involving objects already in the visual field. *Indexing* an object refers to selecting an object that has properties of interest, which Agre called *indexable.* Let's also assume that all moving objects are *marked. Marking* refers to placing a marker on an object or a location for future reference. This is used in tracking a moving object. When the agent decides to execute an act, let's say *grasp(apple)*, if the required types of objects of the act match the type of object thus far identified, the third question is answered by a form of Functional Unification (FU), which finds the utility of the object in the context of the agent's current task. In our example, the argument of *grasp* is of the same type as the object sensed; they are both *apple* types. Using FU, the object at the Knowledge Level is identified indexico-functionally as an-apple-on-the-table-that-I-want-to-grasp. Note that all representations are from the perspective of the agent, so 'I' is automatically integrated into indexical-functional representations. It is at this stage that the agent propagates its decision *to grasp* down to the Perceptuo-Motor Level.

In the second case, the agent has decided to execute *grasp(apple)* but it does not yet know the referent of the argument of the function *grasp()*. The agent must postpone executing its grasping until it situates itself in an environment where it can sense objects of the type matching the object types of the action. This is in effect an active vision task. Assuming the agent has found an object of the desired type, it can proceed as in the first case. The only difference in the second case is that the agent has to first plan and execute finding an object before it can proceed with FU.

2.2.3 Engaged and Disengaged Reasoning

My architecture allows me elegantly to model two different modes of reasoning and planning, which I call *engaged reasoning* and *disengaged reasoning.* Engaged reasoning takes place when all the elementary symbols at the Knowledge Level that are involved in the current reasoning activity are aligned with representations at the Perceptuo-Motor Level. In practical terms, this means that the agent is reasoning about objects within its field of perception. This may require active perception to keep track of objects, or to shift attention to new objects as the reasoning progresses. For the purpose of object tracking and attention shifting, Knowledge Level sensory actions are defined and can be reasoned about like ordinary actions. Engaged reasoning can be done while the actions being reasoned about are actually being carried out, in a kind of plan-as-you-go mode with continuous monitoring of progress being made, or in a more hypothetical mode with no actions being carried out, but potentially affected objects being gauged while the plan is being developed.

Disengaged reasoning occurs when there is no alignment at all between the Knowledge Level symbols involved in the current reasoning activity and Perceptuo-Motor Level representations, e.g., when developing a plan for another place and/or another time, in a purely hypothetical fashion. This is the mode that traditional planners used to operate in all the time, by necessity. Intermediate forms of reasoning, between engaged and disengaged, are possible as well.

2.2.4 Symbol Grounding: A Non-Tarskian Semantics

> Tarskian Semantics has nothing to say about how descriptions of objects in plans relate to the objects in the world [McDermott, 1991, p. 13].

Let's digress for a moment to some esoteric matters of semantics and reference. One problem an agent has to solve is how to find and maintain a correspondence between a referent in the world and a symbol in an agent's world model. The referent in the world is (by necessity) only indirectly considered via its embodied Perceptuo-Motor Level representation; hence, the problem becomes one of aligning the Knowledge Level representations with the Perceptuo-Motor Level representations. From the perspective of cognitive science, the problem has been labeled the *symbol grounding problem* [Harnad, 1990]. The question is how to make the semantics of a robot's systematically interpretable Knowledge Level symbols cohere equally systematically with the robot's interactions with the world, such that the symbols refer to the world on their own, rather than merely because of an external interpretation we place on them. This requires that the robot be able to discriminate, identify, and manipulate the objects, events, and states of affairs that its symbols refer to [Harnad, 1992]. Grounding is accomplished in our architecture in part through the alignment of the Knowledge and Perceptuo-Motor Levels. Elementary symbols at the Knowledge Level are grounded in the sense that they only attach to "the right kind" of representations at the Perceptuo-Motor Level. If we think of the Perceptuo-Motor Level as implementing perception of categories (and perhaps "categorical action"), then the elementary symbols of the

Knowledge Level are the names attached to the categories. In other words, the alignment of the Knowledge and Perceptuo-Motor Level constitutes an *internal referential semantic model* of elementary symbols. Note that, like McDermott, we do not take the Tarskian stance that requires the referents of symbols to be in the world; rather, they are system-internal or what Harnad calls iconic representations: "proximal sensory projections of distal objects, events, and states of affairs in the world" [Harnad, 1990]. The Knowledge Level is the only level that is accessible for "conscious" reasoning, and also the only level that is accessible for inter-agent communication. Access to the Perceptuo-Motor Level and the Sensori-Actuator Level would not be useful for communication, as the representations and processing at these levels are too agent-centered and too agent-specific to be informative to other agents.

Since the Perceptuo-Motor Level representations serving as the grounding for symbols of the Knowledge Level are embodied (section 2.2.5), equivalent symbols may have somewhat different semantics for different agents having different bodies. We don't see that as a problem, as long as the differences are not too large.[11] Indeed, we believe that this is quite realistic in human terms as well; no two persons are likely to have *exactly* the same semantics for their concepts, which nevertheless does not prevent them from understanding each other. The problems of translation and communication in general consist at least in part of establishing a correspondence between concepts (and symbols) used by the participants. It is helpful to be able to use referents in the external world as landmarks in the semantic landscape, but one consequence of embodied semantics is that *even* if it is possible to establish these common external referents for symbols, there is still no guarantee that the symbols will actually *mean* exactly the same thing, because in effect the same referent in the world is *not* the same thing to different agents. If we accept this view, it is clear that approaches to semantics based on traditional logical model theory are doomed to fail, because they *presuppose* "identity of referents" and an unambiguous mapping from symbols to referents, the same one for all agents. Another problem is of course the presupposition that all objects are uniquely identifiable. The use of deictic representations does not impose such a condition; as far as our agents are concerned, if it looks and feels the same, it is the

[11] It is never a problem as long as agents need not communicate with the outside world (other agents), of course, cf. [Winston, 1975].

same.[12] Nothing hinges on whether or not the objects in the agent's surroundings are *really* extensionally the same as the identical-looking ones that were there a moment ago or will be there a moment later.

2.2.5 Embodied Representation

In section 2.2.1.3, I mentioned the use of embodied representations at the Perceptuo-Motor Level. I now look at the principle of embodiment from a more abstract point of view.

One of the most general motivations behind our work is the desire to be able to "program" a robotic autonomous agent by requesting it to do something and have it "understand", rather than telling it how to do something in terms of primitive motions with little or no "understanding". For instance, we want to tell it to go find a red pen, pick it up, and bring it to us, and not have to program it at a low level to do these things.[13] One might say that we want to communicate with the robot at the *speech act* level. To do this, the agent needs a set of general-purpose perceptual and motor capabilities along with an "understanding" of these capabilities. The agent also needs a set of concepts which are similar enough to ours to enable easy communication. The best way to accomplish this is to endow the agent with concepts that are embodied (according to the agent's body) and grounded in perception and action.

We define *embodiment* as the notion that the representation and extension of high-level concepts is in part determined by the physiology (the bodily functions) of an agent, and in part by the interaction of the agent with the world. For instance, the extension of color concepts is in part determined by the physiology of our color-perception mechanism and in part by the visual stimuli we look at. The result is the establishment of a mapping between color concepts and certain properties of both the color perception mechanism and objects in the world. Another example is the extension of concepts of action: it is partly determined by the physiology of the agent's motor mechanisms and partly by the interaction with objects in the world. The result is the establishment of a mapping between concepts of action and certain properties of both the motor mechanisms and objects in the world (what we might call "the shapes of acts").

[12] This is of course the "duck test", made famous by a former US president.

[13] Retrieving "canned" parameterized routines is still a low-level programming style that we want to avoid.

At an abstract level, the way to provide an autonomous agent with human-like embodied concepts is to intersect the set of human physiological capabilities with the set of the agent's potential physiological capabilities, and endow the agent with what is in this intersection. To determine an agent's potential physiological capabilities, we consider it to be made up of a set of primitive actuators and sensors, combined with a general purpose computational mechanism. The physical limitations of the sensors, actuators, and computational mechanism bound the set of potential capabilities. For instance with respect to color perception, if the agent uses a CCD color camera (whose spectral sensitivity is usually wider than that of the human eye), combined with a powerful computational mechanism, we consider its potential capabilities wider than the human ones, and thus restrict the implemented capabilities to the human ones. We endow the agent with a color perception mechanism whose functional properties reflect the physiology of human color perception. That results in color concepts that are similar to human color concepts. With respect to the manipulation of objects, most robot manipulators are inferior to human arms and hands; hence, we restrict the implemented capabilities to the ones that are allowed by the robot's physiology. The robot's motor mechanism then reflects the properties of its own physiology, rather than those of the human physiology. This results in a set of motor concepts that is a subset of the human one. Embodiment also calls for body-centered and body-measured representations, relative to the agent's own physiology. We provide more details on embodiment in GLAIR in [Hexmoor et al., 1993d].

2.2.6 Alignment

When a GLAIR-agent notices something in its environment, it registers that it has come to know of an object. Regardless of whether the agent recognizes the type of the object, we want it explicitly to represent the existence of the object in the Knowledge Level while processing sensory information about the object in the Perceptuo-Motor Level. Similar to when sensing objects, when properties of objects or relationships among objects are sensed by the GLAIR-agent, we want it to explicitly represent these properties and relationships, even if no more is known about them than the fact that they exist. We use unnamed intensional concepts for this purpose [Shapiro and Rapaport, 1986].

Having sensed an object, an assertion is made about the object being sensed in the

GLAIR Knowledge Level. Once the object is no longer in the "field of perception", the assertion about its being sensed is removed. This is tantamount to disconnecting the relationship between the symbolic representation and the world. If at the Perceptuo-Motor Level a previously sensed object is again being sensed, we reassert the fact that the object, the same one represented before in the Knowledge Level, is being sensed. An example of this type of ("unconscious") perception is when we look at an object, look away, and then look back at the same object. The "unconscious" level can provide a short term sensory memory in which memories of objects are stored, and when we see them from time to time, the "conscious" layer is alerted to that fact. We can think of this phenomenon as a type of *continuity in perception* at the "unconscious" level. If we assume this continuity, we should re-use previously constructed representations to represent again-sensed objects. In order for a GLAIR-agent to re-use its previously established representations about objects for again-sensed objects, we either have to assume that the agent has a continuity of perception at the "unconscious" layer or that a "conscious" matching of existing representations to sensed objects is performed. For a detailed discussion of alignment, see [Caicedo and Hexmoor, 1993].

2.2.7 "Consciousness"

As I pointed out above, we identify the Knowledge Level with "consciously" accessible data and processing; the Perceptuo-Motor Level with "hard-wired", not consciously accessible processing and data involved with motor control and perceptual processing; and the Sensori-Actuator Level with the lowest-level muscular and sensor control, also not consciously accessible. The distinction of "conscious" (Knowledge) levels vs. "unconscious" (Perceptuo-Motor and Sensori-Actuator) levels is convenient as an anthropomorphic metaphor, allows us to separate explicitly represented and reasoned about knowledge from implicitly represented and processed knowledge. This corresponds *grosso modo* to consciously accessible and not consciously accessible knowledge for people.[14] Although we are aware of the pitfalls of introspection, this provides us with a rule of thumb for assigning knowledge (and skills, behaviors, etc.) to the various levels of the architecture. We believe that our organization is to some extent psychologically relevant, although we have not yet undertaken any exper-

[14]The term "knowledge" should be taken in a very broad sense here.

imental investigations in this respect. The real test for our architecture is its usefulness in applications to physical (robotic) autonomous agents (chapter 6).

Knowledge in GLAIR can migrate from "conscious" to "unconscious" levels. In [Hexmoor et al., 1993a] I show how a video-game playing agent learns how dynamically to "compile" a game playing strategy that is Initially formulated as a set of explicit reasoning rules at the Knowledge Level into an implicit form of knowledge at the Perceptuo-Motor Level, a Perceptuo-Motor Automaton (PMA).

There are also clear computational advantages to our architectural organization. A Knowledge Representation and Reasoning system as used for the "conscious" Knowledge Level is by its very nature slow and requires lots of computational resources.[15] The implementation mechanisms we use for the "unconscious" levels, such as PMAs, are much faster and require much fewer resources. Since the three levels of our architecture are semi-independent, they can be implemented in a (coarse-grained) parallel distributed fashion; at least each level may be implemented on distinct hardware, and even separate mechanisms within the levels (such as individual reflex behaviors) may be. Our Robot Waiter agent, for instance, uses distinct hardware for the three levels. See our chapter on GLAIR agents for further details.

2.3 Background on Architectures

I review architectures that influenced the development of our architecture and contrast our architecture with others. In the few years that we have been developing our architecture, many researchers have enhanced their architectures. A review of architectures becomes dated in a few months, almost in the time it takes to write about them. In this dissertation, my review will cover my review of architectures until March 1995 at the time when I co-chaired a AAAI symposium on architectures.

Architectures proposed in the literature do not fall into neatly separable classes, mainly because the scope of the models and the motivations varies widely. However, we can divide a review of related work into theoretical issues of agent architectures on the one hand, and

[15] Many reasoning problems are NP-complete, meaning there are no polynomial-time deterministic algorithms known for solving them, or in plain English: they are very hard to solve in a reasonable amount of time (see e.g. [Hausser, 1989]).

implemented architectures on the other.

2.3.1 Theoretical Issues

I believe that behavior-based AI has adopted the right treatment of everyday behavior for agents that function in the world. However, this has been done at the expense of ignoring cognitive processing such as planning and reasoning. Clearly, what is needed is an approach that allows for both. I believe that our architecture meets this need. As in behavior-based AI, GLAIR gains validity from its being grounded in its interaction with the environment, while it benefits from a knowledge level that, independent of reacting to a changing environment, performs reasoning and planning.

The Model Human Processor [Card et al., 1983] is a cognitive model with *perception*, *cognition*, and *motor* components. Cognition consists of working memory, long-term memory, and the cognitive processor. Perception is a hierarchy of sensory processing. The motor component executes the actions in the working memory. This is a traditional symbol-system decomposition of human information processing. This type of decomposition has shown only limited success in building physical systems. These systems are limited in their (a) lack of representational capability to model closed-chain monitoring loops, e.g., feedback loops, (b) ability to model short reaction times, and (c) lack of representational power to model quantitative data—such as the inability to model strength of connections between concepts in neural networks. These systems have a top-down organization in that all sensory data travels through the perceptual mechanism, reaches the cognitive levels, and finally the motor component. The chain of sensation to action generation is a top-down mechanism. Systems like SOAR adhere to this model. In our architecture, we purposely avoid this kind of top-down problem decomposition by allowing independent control mechanisms at different levels to take control of the agent's behavior and pre-empt higher-level control while doing so. It may be necessary to allow higher-level mechanisms selectively to inhibit lower-level ones as well. For example, for a robot with a bumper sensor, a normal reflex would be for the robot to withdraw from collisions when the bumper is depressed. However, if we want this robot to push a block, the reflex to withdraw needs to be inhibited so the robot can push the block while the bumper is depressed. By providing subsymbolic layers and semi-independent processes that interact with the world relatively quickly, the organization of

GLAIR levels and processes addresses the limitations we pointed out in competing systems.

A situated agent, at any moment, attends only to a handful of entities and relationships in its immediate surroundings. In this type of setting, the agent often does not care to uniquely identify objects. It is sufficient to know the current relationship of the relevant objects to the agent and what roles the objects play in the agent's activities. Agre and Chapman (in [Agre and Chapman, 1987]) proposed indexical-functional representations (which [Agre, 1988] refers to as deictic representations) to be the more natural way agents refer to objects in common, everyday environments. They called entities and relationships of interest *entities* and *aspects*, respectively. With respect to its current activities, the agent needs only to focus on representing those entities and aspects. Although the objects in the environment come and go, the representations of entities and aspects remain the same. For example, the-cup-that-I-am-holding[16] is an indexical-functional notation that abstracts the essentials of what the agent needs to know in its interaction. These representations serve to limit the scope of focus on entities. For example, if the agent wants to pick up a cup, it does not need to know *who owns the cup* or *how much coffee the cup can hold*; only the relevant attributes of the cup apply. We believe that systems endowed with general knowledge representation and reasoning abilities can and should generate deictic representations to create and maintain a focus on entities in the world and we have designed an implementation strategy in our chapter 6.

2.3.2 Implemented Architectures

RCS is an example of a hierarchical control architecture [Albus, 1995, Albus et al., 1981] and an example of a robotic architecture; we would say it is *body centered.* This architecture proposes abstraction levels for behavior generation, sensory processing, and world modeling. Lower in the hierarchy, tasks are decomposed into robot-motion primitives. This differs from GLAIR, which is not strictly top-down controlled. In RCS, concurrently at each level of the hierarchy, feedback processing modules extract the information needed for control decisions at that level from the sensory data stream and from the lower level control modules. Extracted environmental information is compared with the expected internal states to find

[16]This kind of designation is merely a mnemonic representation intended to suggest the entity and aspect under consideration, for the purpose of our exposition. It is not the actual representation that would be used by an agent.

differences. The differences are used for planning at higher levels.

Bonasso and Kortenkamp's 3T [Bonasso and Kortenkamp, 1995] is very similar to GLAIR in functionality of levels. 3T is composed of three levels: planning, sequencing (adopted from Jim Firby's RAPs system [Firby, 1987]), and skills. 3T makes provisions for robots that take instruction from a human supervisor while the robot performs tasks autonomously and avoids danger. This system is similar to GLAIR in having three levels and in the types of intended applications. I believe GLAIR's KL and PML map into 3T's three levels. 3T does not have a level corresponding to GLAIR's SAL.

Brooks's *subsumption* architecture [Brooks, 1985, Brooks, 1987, Brooks, 1990] clusters behaviors into levels. Low-level behaviors, like deciding the direction of motion and speed, can be inhibited (subsumed, taken over) by behaviors determined at higher levels, such as avoiding obstacles. Subsumption behaviors are written as finite-state machines augmented with timing elements. A compiler is used to simulate the operation of finite-state machines and parallelism. This architecture is implemented on a variety of mobile robots. Frequently-used behaviors are placed at a lower level than less frequently-used behaviors. This organization of behaviors gives the system fast response time and high reactivity. Our architecture is similar to Brooks's in our intra-level implementations of behaviors. However, the subsumption architecture lacks the separation we have made into "conscious" and "unconscious" spheres. In anthropomorphic terms, Brooks's agents are all "unconscious". We believe that the off-line specification and compilation of behavior modules is too inflexible for autonomous agents that can adapt to a wide range of circumstances, especially if they have to learn from their interactions with the environment.

Pattie Maes has experimented with a version of a behavior-based architecture, which she calls ANA [Maes, 1991]. This architecture consists of competence modules for action and a belief set in a network relating modules through links denoting successors, predecessors, and conflicts. Competence modules have activation levels. Activations are propagated and the competence module with the highest activation level is given control. Maes has explored learning and has applied her architecture to robotic systems. ANA is similar to PMA in GLAIR. ANA and Maes's experiment with an insect robot that learns to walk is an impressive demonstration of action selection and a physical system that learns. However, this is only part of the functionality of an architecture for intelligent agency such as GLAIR;

other functionalities, like mechanisms that model bodily reflexes or that are unlearnable, are needed.

In the subsumption architecture, sensations and actions are abstracted by giving them names like "straightening behavior" in order to make things easier to understand for human observers. Much in the spirit of [Agre, 1988], we believe that behavior modules should more naturally emerge from the interaction of the agent with its environment. In contrast to hand coding behaviors and in order to facilitate embodiment, in GLAIR we are experimenting with unnamed emergent behavior modules that are learned by a robot from scratch. A behavior module can be thought of as a set of pairs (P,A), where P is a set of grounded sensations and A is an instance of an act. For instance, reaching for an object might be a set of pairs (vision/sonar data, wheel motor actuation). After learning, this new behavior module will become active only if the grounded sensations match any of the grounded sensations experienced before. As a measure of abstraction and generalization, we may allow near matches for sensations. To bootstrap the learning process, we need a set of primary or first-order ("innate," for the philosophically inclined) sensations and actions.

The Servo, Subsumption, Symbolic (SSS) architecture [Connell, 1992] is a hybrid architecture for mobile robots that integrates the three independent layers of servo control (closed feedback loop between sensing and acting), Brooksian behavior-based modules, and a symbolic layer. Our architecture is similar to this in its general spirit of identification and integration of three distinct levels corresponding to levels of affinity-of-interaction with the world (i.e., responsiveness to the events in the world). This similarity also constitutes a point of departure, however, in that SSS is defined with respect to specific (and different) implementation techniques. For example, the symbolic layer in SSS is a decision table instead of a general knowledge representation and reasoning (KRR) system as used in GLAIR. Unlike GLAIR, SSS assigns particular tasks for each layer and uses a hard-wired interconnection channel among layers. We are interested in flexible connections, because we would like these connections to be subject to learning.

[Brady and Hu, 1995] report on industrial mobile robots with modular units they call Locally Intelligent Control Agents (LICAs). Each LICA has its own local knowledge, decision making, and is responsible for communication with other LICA. Each robot is programmed with about a dozen LICAs. This is an interesting decentralization. However, it is not clear how LICA organization addresses cognitive processes or the import of LICA in intelligent agency.

Payton (in [Payton, 1986]) introduced an architecture for controlling an autonomous land vehicle. This architecture has four levels: mission planning, map-based planning, local planning, and reflexive planning. All levels operate in parallel. Higher levels are charged with tasks requiring high assimilation and low immediacy. The lower levels operate on tasks requiring high immediacy and low assimilation. Our architecture is similar in this respect. The reflexive planning is designed to consist of pairs of the form ⟨virtual sensor, reflexive behavior⟩. Each reflexive behavior has an associated priority, and a central blackboard-style manager arbitrates among the reflex behaviors. Some of the problems with the earlier implementation due to using the blackboard model were solved in [Rosenblatt and Payton, 1989].

Gat (in [Gat, 1991]) describes ATLANTIS, an architecture for the control of mobile robots. This architecture has three components: control, sequencing, and deliberation. The control layer is designed as a set of circuit-like functions using Gat's language for circuits, ALPHA. The sequencing is a variation of Jim Firby's RAP system [Firby, 1987]. The deliberation layer is the least described layer. Structurally and functionally, our system is similar to Gat's. I have attempted to map my architectural levels closes to psychologically understood functionalities.

An architecture for low-level and high-level reactivity was suggested in [Hexmoor, 1989]. High-level reactivity is reactivity at the conceptual level. This architecture suggests that an autonomous agent maintains several different types of goals. High-level reactivity is charged with noticing impacts of events and actions in the environment on the agent's goals. Subsequently, high-level reactivity needs to guide the agent's low-level reactivity. Low-level reactivity is at the sensory, perceptual, and motor level. The mechanism for low-level reactivity is similar to other reactive systems that have components for perception and action arbitration. The novelty of this architecture is the incorporation of high-level reactivity and a supervisory level of planning and reasoning, which guides the choice of low-level reactive behaviors. In our present conception of agent architecture, we avoid a sharp separation between the two types of reactivity. We also relax the top-down nature of interaction between levels. Reactivity may be initiated at any level of our architecture either due to interaction with other levels or in direct response to external stimuli.

Simmons's Task Control Architecture (TCA) [Simmons, 1990] interleaves planning and acting by adding *delay-planning constraints* to postpone refinement of planning until execution. For example, a plan for a robot to collect used cups for trash is decomposed into: navigate to the cup; pick it up; navigate to trash bin; deposit the cup. Since the robot does not have sensory information about the cup yet, the plan to pick it up is delayed until the robot gets close enough. Selectively delaying refinement of plans allows for reactivity. TCA uses a message-passing scheme among modules that allows concurrent execution of tasks. It has been used to control "Ambler", a six-legged walking robot and another cup-collecting robot. TCA's "stepwise refinement" follows effortlessly from GLAIR, without the need to implement it explicitly. This is because our architecture has concurrent processes and refinement is not centrally controlled. Since "conscious" planning that goes on at the Knowledge Level uses a more coarse-grained world model, it is impossible to express fine details of planning and execution at that level. These can only be represented or computed at the lower Perceptuo-Motor and Sensori-Actuator levels. Planning and execution in our architecture may proceed in a lock-step fashion, but they need not do so.

SOAR [Laird et al., 1987] was designed to be a general problem-solving architecture. SOAR integrates a type of learning known as *chunking* in its production system. Recently, SOAR has been applied to robotic tasks [Laird et al., 1991]. In this framework, planning and acting are uniformly represented and controlled in SOAR. In my approach, controlled, "conscious" processes are traded for automatic "unconscious" processes [Shiffrin and Schneider, 1977]. Soar does not make the distinction between "unconscious" and "conscious" processes.

Chapter 3

Routines

The study of routines takes us most directly to the literature in the fields of planning, artificial life, integrated approaches and architectures, and various psychological studies on automaticity, chunking, microgenetic study, motor control, and action slips. Since my work is in the general school of *reactive planning*, I will focus my planning review on reactive planning and compare my approach with other work in this area. In addition to this, I briefly review the learning literature and contrast it with what I find necessary for learning routines.

3.1 Planning

Reactive planning takes the situated-action view of planning. That is, the agent decides on an action based on the current situation. This is in contrast to classical planning where effects of actions, mental states, and historical events may be considered. All reactive systems share the following properties:

- Applicability of only one action is decided upon at any moment. Actions are not queued or assigned a status of readiness for execution.

- No predictions are made about what will be true after completing an action.

Since reactive planners make no predictions about the results of actions, the concepts of plan failure and replanning are avoided. With other planners, a plan fails if either a plan step becomes inappropriate with respect to the conditions in the environment or the post-conditions of a plan step after execution are not true in the environment. A planner that fixes a failed plan is said to be replanning. The reactive control strategy of James Firby's RAPs helps it to avoid replanning. Firby states, "...execution monitoring is an intrinsic part of the execution algorithm, and the need for separate replanning on failure disappears." [Firby, 1987, p. 203]

3.1.1 Reacting as Improvisation

Agre and Chapman [Agre and Chapman, 1990] have championed acting as emergent interactions between an agent and its environment. Pengi and Sonja are two computer video games they developed that exemplify reacting as improvisation. In these games, the game-hero engages in sophisticated interactions with the opponent without any prior planning. However, the strategies the hero exhibits appear to be planned.

Reacting as improvisation emphasizes the following properties:

- Ready to Act assumption: The planner is always ready to interact with the environment.

- Deictic Representations assumption: An agent interacting in its environment only needs to represent entities in terms of how they are related to the current task and how they are related to the agent.

- Lack of Control assumption: "...abandon the notion that activity has a particular locus of control at all. Activity arises through interaction, not through control." [Agre and Chapman, 1990, p. 22].

I have developed GLAIR as an architecture for an agent [Hexmoor et al., 1992]. In GLAIR, I have developed a mechanism for a kind of "unconscious" behavior generation I call Perceptuo-Motor Automata (PMA). My approach in GLAIR and PMA is consistent with work by Agre and Chapman. GLAIR and PMA are computational techniques designed to study improvisations and learning commonplace in routine activity.

3.1.2 Reacting as a Stateless Device

A reactive planner is a system which reacts as a stateless device (a system that does not have internal states). It generates plans that the agent puts off the decision, not the plan to react until execution. Such a planning system is considered to be an automatic constructor of reaction plans. Here, the reactivity lies in the plan and not in the planning strategy. After Marcel Schoppers [Schoppers, 1987], I'll call this type of reactive planner a "reactive reaction planner," abbreviated as RRPlanner. Schoppers's universal planning has an RRPlanning component. RRPlanning does not interact with the environment; rather, it is the plan interpreter that interacts with the environment. In this respect, building a reaction plan is like building a device. Reaction plans are an extreme approach to acting with little or no thinking, i.e., purely "unconscious". This might be suitable for general-purpose agents in some activities or for some insect-like agents. These planners share the following properties:

- Situatedness assumption: Applicability of an action is decided based on the currently perceived situation and not on the analysis of the history of what has happened.
- Infallible reactions assumption: Interference among actions is computed, and situated rules are designed to steer away from undesirable states.[1] This is also known as the *causal independence assumption.*

In Schoppers's Universal Planner, an action is represented as a collection of effect description rules.

> In general, effect description rules have the form
>
> action "=" [*qualifiers* "?"] [*subgoals*] $\rightarrow$ *a-goals* ["#" *m-goals*]
>
> and the components have the following roles:
>
> - *action* is the name of the action being described.
> - *a-goals* is a conjunction of one or more conditions such that the entire conjunction can be made true by the action. These conditions are the goals of achievement for which the action is useful.

[1] By interference among actions I mean that the effect of one action is undone by another action.

- *subgoals* is a conjunction of conditions that must be achieved before performing the action. When omitted, *subgoals* is interpreted as *true*.
- *qualifiers* is a conjunction of conditions that restricts the applicability of the rule: this rule cannot be used unless all the specified conditions are true in the world. When omitted, *qualifiers* is interpreted as *true*.
- *m-goals* is a conjunction of the conditions that restricts the applicability of the rule: rule cannot be used unless all the specified conditions are goals of maintenance. When omitted, *m-goals* is interpreted as true.

[Schoppers, 1989, p. 13]

An example is the action of *closing* a robot hand

null $\longrightarrow$ *not wide*,

which means keep your hand closed by default (*wide* refers to a two fingered robot hand in the position of fingers being wide apart), and

around(X) $\longrightarrow$ *holding(X)*,

which means if you want to want to be holding something, hand must be around that thing, as two goal-reduction rules for this example. Maintenance goals are nonexistent in both of these rules. Q in the second rule quantifies X to be all objects that the robot can get its hands around. In Schoppers's representation, he explicitly omits putting Q in the rule.

A universal plan for a goal is a decision tree that is derived by backchaining through action descriptions. If a condition is false at a node of the decision tree, then it is false in all subtrees below that node as well. The universal planner generates and uses *confinement rules*. These rules are constraints of causal interference among actions. Figure 3.1 is a diagram of the universal planner.

Universal plans are necessary and sufficient representations of an agent's know-how in reactive systems. I make this judgment based on the criteria for reactive systems that I stated earlier. Universal plans and Schoppers's mechanisms to convert plans to universal plans is similar to my PMA and my algorithm for conversion of plans into PMA (see [Hexmoor and Nute, 1992]) (for details).

Drummond's Situated Action Rules is another example of reacting as a stateless device. [Drummond, 1989] describes a planning system that generates Situated Control Rules

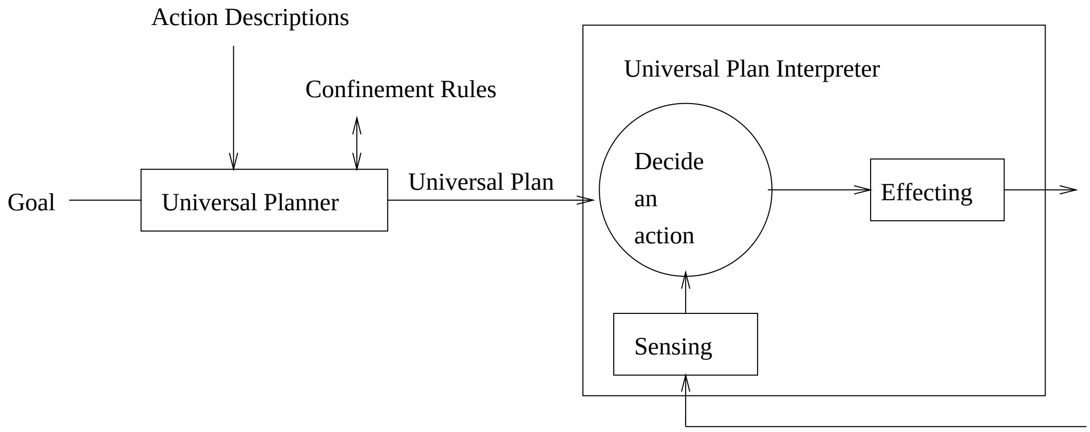

Figure 3.1: Universal Planning: Goals and action descriptions are used to generate a universal plan. Confinement rules are constraints that are either used in the process of universal plan generation or discovered in this process. Universal plan interpreter senses the environment, picks an action using the universal plan and sends it to mechanisms for effecting (i.e., executing) the action.

(SCRs) from *plan nets*. Plan nets are intended to be a language for specifying basic capabilities of an agent. A plan net is a bipartite graph of conditions and events. Directed edges pointing from events to conditions are *enablement edges*, and edges between conditions and events are *causal* edges. Through a process of *projection*, SCRs are generated from plan nets. SCRs are rules that embody causal knowledge about the domain. SCRs are used to guide action selection and to steer away from undesirable states.

In my approach, a PMA is a reaction plan that can be compiled from a knowledge level plan. In fact, in [Hexmoor and Nute, 1992] I use a PMA for the main example of [Drummond, 1989]. In addition to PMA being representationally as powerful as Drummond's SCR and Schoppers's universal plans, I have developed an algorithm for automatic conversion of output of a classical planner, which are plans, into PMA [Hexmoor and Nute, 1992]

Brooks's subsumption architecture clusters behaviors into layers [Brooks, 1985, Brooks, 1987, Brooks, 1990]. Low-level behaviors, like deciding the direction of motion and speed, can be interrupted by behaviors determined at higher levels, such as avoiding obstacles. Subsumption behaviors are written as finite state machines augmented with timing elements. A compiler is used to simulate the operation of finite state machines and parallelism. This architecture is implemented on a variety of mobile robots. More often used behaviors are placed at a lower level than less frequently used behaviors. This organization of behaviors gives the system faster response time and higher reactivity.

Maes has experimented with a version of a behavior-based architecture that she calls ANA [Maes, 1991]. This architecture consists of competence modules of the form [Activation level, Conditions, Add-list, Delete-list, Body of module]. These modules are placed in a network relating modules through links denoting successors, predecessors, and conflicts. Competence modules have activation levels. Activations are propagated, and the competence module with the highest activation level is given control. Achievement goals and *protected* goals are used in networks of competence modules to change and propagate activation levels. I will review Maes's reinforcement-based learning later in this dissertation.

An underlying assumption in subsumption architectures is that all tasks are automatic, i.e., operating at the non-cognitive level. Operations of processing modules may seem to be like an electronic circuit, fast and without reason or explanation. Computational models of interaction like Brooks's subsumption are fine for the tasks to which they are applied. However, they do not provide a uniform model of interaction at various cognitive levels. This was addressed in my model for intelligent agency, GLAIR, which I presented in chapter 2.

3.1.3 Reacting as an Automaton

Kaelbling [Kaelbling, 1986, Kaelbling, 1988] describes a reactive architecture for robot planning. This architecture defines a hierarchical organization of robot behaviors and a control scheme that determines the appropriateness of available behaviors. Kaelbling and Rosenschein's work [Kaelbling, 1988, Kaelbling and Rosenschein, 1990] describes tools (REX, GAPP, RULER) that, given descriptions of the world and tasks, construct reactive control mechanisms termed *situated automata*. Their architecture consists of perception and action components. The robot's sensory input and its feedback are inputs to the perception component. The action component computes actions that suit the perceptual situation. I note that unlike Brooks's behavior modules, situated automata use internal states. Thus, their decisions are not Markovian. They are mainly intended to produce circuits that operate in real time, and they can prove properties about their operation. [Rosenschein and Kaelbling, 1986] defines a model for a deterministic automaton for an agent (planner) and a similar model for the environment. This analysis of the interaction between the two automata relied upon possible-worlds semantics of a modal knowledge operator. The mechanism for generating situated automata, although impressive, seems too inflexible for autonomous agents that have to operate in a wide variety of (possibly unknown) circumstances . Perhaps the operation of my Perceptuo-Motor level could be modeled by a situated automaton, but I am not convinced that this is the right formalism to use, due to its inflexibility.

Rosenchein and Kaelbling's work is similar in spirit to my own in the way that a state machine is generated. It is also similar to my work in that they consider an internal state. Like situated automata, my PMA is a specification language for *machines*. My motivation in developing PMAs was to build "reaction plans" from classical plans. This has led me to develop a particular type of automaton in which the effects of actions are not used in an operating PMA even though the effects of actions are used in constructing PMAs. However, situated automata are designed to build a machine's behavior where behaviors can be described in terms of logic circuits. This leads to automata in the classical sense, in that the effects of actions are used as inputs to actions, and they are used when a situated automaton is operating.

3.2 Artificial Life

> Artificial life, or a-life, is devoted to the creation and study of lifelike organisms and systems built by humans. The stuff of this life is nonorganic matter, and its essence is information: computers are the kilns from which these new organisms emerge. Just as medical scientists have managed to tinker with life's mechanisms in vitro, the biologists and computer scientists of a-life hope to create life *in silico*. [Levy, 1992, p. 5].

A-life has attracted scientists from varying disciplines. What is in common is the creation of creatures that appear lifelike. These creatures "live" and may "learn," "reproduce," and "die." Many of these creatures are given habitats and are studied for their behavior and evolution. It is yet too early in A-life to identify common techniques for building creature or studying them. However, there is a general emphasis to produce physical systems that interact in realistic situations. There is a trend toward building inexpensive and simple robots with very simple behavior engines and embedding them in a physical environment. In so doing, external observers note interacting behaviors that make the agent appear intelligent and unexpected. These behaviors are known as emergent behaviors.

> *Emergent behavior* is one of the main topics of research in the field of swarm intelligence. Such behavior is characterized by the following properties: 1) it is manifested by global states or time extended patterns which are not explicitly programmed in, but result from local interaction among a system's components, and 2) it is considered interesting based on some observer-established metric. [Mataric, 1993, p. 433]

In my investigation of routines, I am also striving for creatures that inhabit a complex world and interact with it. Whenever I can, I try to build physical agents. My interest in emergent behaviors is in trying to get the agents to notice and make use of emergent behaviors. This differs from an observer study of such behaviors.

3.3 Integrated Approaches and Architectures

In this section, I briefly cite literature for integrating planning and acting and suggestions made in the literature for architectures for intelligent agency. An architecture is an organization of components and interconnections among components for systems (agents) that addresses the problem of integration. In contrast to GLAIR, these architectures do not consider separate layers for "conscious" and "unconscious" mechanisms of an agent, which I consider central to learning routine activity. GLAIR's learning is similar to [Mitchell, 1990] in general, but I consider reinforcement-based learning, whereas Mitchell used explanation-based learning.

Recent literature is full of suggestions for integrating different planning paradigms mediated by learning. The RAP system [Firby, 1987] attempts to integrate goal-directed reasoning and reactive behavior. A RAP plan consists of a list of tasks instead of actions. A RAP consists of three components: a satisfaction test, a window of opportunity, and a set of context-dependent methods. Tasks are invoked based on system beliefs about the current world situation. Then these tasks are expanded hierarchically, and appropriate methods are selected for execution. Figure 3.2 shows a simplified architecture for RAP systems. Since it reinvokes already tried task networks, it encounters the problem of indefinite looping, i.e., trying to do a task that might be undoable.

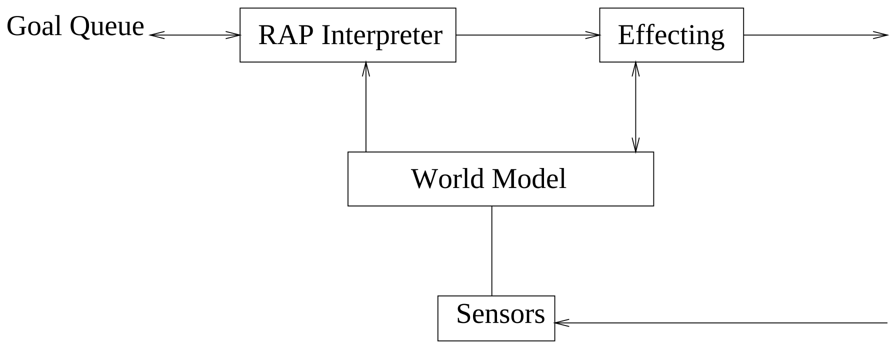

Figure 3.2: A Simplified Schematic of RAP Planning: RAP sensors sense the environment and update the world model. The world model is used in picking a goal from the goal queue. Once a goal is selected, a method is sent to effecting mechanisms. The world model is used in instantiating actions. Finally, the world model is updated with information about execution of actions.

Mitchell and his group are developing agents that integrate planning and learning [Blythe and Mitchell, 1989, Mitchell, 1990, Mitchell et al., 1991]. Theo-Agent [Mitchell, 1990] is a reactive architecture that integrates a few types of learning with reacting, and in the process its decision making becomes faster. Mitchell's agents are Initially classical planners. Using explanation-based learning (EBL), and a few other techniques, agents form new stimulus-response rules to use in the future. Using learned rules, agents become faster.

Another system that starts as a classical system and learns reactive rules using reasoning is that of [Gervasio and DeJong, 1989, Gervasio, 1990]. Gervasio suggests a form of EBL which she calls contingent EBL (CEBL). CEBL is shown to work on learning reactive rules over continuously changing situations. The key point of CEBL introduces *conjectured variables*. These are variables that do not have exact values until execution.

In my own early work on architectures [Hexmoor, 1989, Hexmoor et al., 1990], I considered categories of high-level goals such as achievement, preservation. I demonstrated that reactive action selection is guided and interrupted by changes in the high-level goals.

I argued that reactivity should be considered in deciding *what to do* as well as in deciding on *higher mental activities*.

Since my architecture relies on integrating planning and reacting, I have developed a new approach and an architecture for building an intelligent agent that provides a framework for studying routine activities. By providing several layers, we can classify an agent's know-how and examine how the nature of know-how and thereby representations and computational mechanisms change over time. For a broad review of planning literature and discussion of my approach to planning, see [Hexmoor and Nute, 1992, Hexmoor et al., 1992].

3.4 Learning

Learning is an important feature of an agent that has to interact in the world. Since I am not interested in domain-specific agents, I would like the agent to be able to cope with what it finds in the world. I am particularly interested in agents that learn to extend their abilities to interact with the world. For example, if all an agent knows is picking up square boxes by using its two fingers to squeeze on opposing faces of a square box and lifting it, and now it faces a round ball, it has to learn either to use more fingers or use its two fingers to squeeze on diametrically opposing points of the ball. An agent that is a novice in performing or choosing an act consciously reasons about the act and controls its execution. As the agent acquires more experience with an act, it is able unconsciously to pick the act and execute it. In psychological terms, this is called *automaticity*. Here I am primarily concerned with learning knowledge of acts and knowledge of using acts. I omit review of broader learning techniques.

Learning can stem from perception of an example, as in explanation-based learning; perception of frequency of a recurring pattern of interaction; or perception of rewards for actions, as in reinforcement-based learning. Learning can be *reflexive* in that the agent has no choice in when it learns, as in Soar's chunking [Larid et al., 1986], or *deliberative*. Learning can be based on externally provided samples or instructions, or can be purely introspective. Learning may be incremental or one-shot.

Macrops [Fikes et al., 1972] was the earliest approach to learning in planning. Explanation Based Learning (EBL) is a descendant of Macrops style learning. EBL is limited to learning knowledge that is within the agent's deductive closure [Russell, 1991]. Recently, Reinforcement Learning (RL) has had a revival in AI [Sutton, 1988] and has been applied to planning [Whitehead, 1992, Kaelbling, 1990, Lin, 1992]. Unlike EBL, RL is used to learn new knowledge that can be beyond the agent's deductive closure. In my approach, I am using and extending RL. RL as a learning mechanism does not make use of prior knowledge. I have developed techniques for learning from prior knowledge and revising the prior knowledge.

3.4.1 What is Learned

In the context of planning, an agent may learn one or more of the following: (1) preconditions (physical and epistemic) to when an act is applicable; (2) post-conditions or effects of actions; (3) new acts; (4) local strategies for applying actions; (5) recurring sequences of actions, i.e., schemas; and (6) decisions for optimal and reliable selection of actions.

In learning the first three things, we may assume that the agent does not have a *complete and correct* theory of its acts. In this case, the learning agent is gaining domain knowledge. Furthermore, an agent that learns new acts is learning new concepts. If the agent assumes it has a *complete and correct* theory of its acts, it may still learn pre- and post-conditions by learning new relationships among its acts. In explanation-based learning, an agent may learn preconditions to its acts that are both derivable from its current theory of acts and supported by an example.

The last three items in the above list most often concern learning domain knowledge about how to improve arbitration among applicable actions. RL combines learning pre- and post-conditions with local strategies and decisions for optimal/reliable selection of actions (i.e., policies, in RL terms).

3.4.2 Analytic Learning

Analytic learning is learning by reasoning. Analytic learners are typically cited in systems with rich domain knowledge. The bulk of work in analytic learning has converged on a technique known as explanation-based learning [DeJong and Mooney, 1986, Mitchell et al., 1986]. EBL applications are found in [Segre and DeJong, 1985, Minton, 1988, Blythe and

Mitchell, 1989, Gervasio and DeJong, 1989]. Explanation-based learning is learning from a single example.

A strong assumption in EBL is the assumption that the domain theory is complete and correct. This makes EBL unsuitable for applications where agents learn from their interaction with the environment. There are analytic learning techniques that fall outside EBL. One example is work reported in [Christiansen et al., 1990]. This work describes a tray with blocks on it handled by a robot. A camera observes the state of the tray. In a loop, the robot forms plans for goals to have the blocks slide in a certain configuration by tilting. However, unlike EBL, the theory of tilting is assumed to be subject to errors. With each tilt, the robot's theory of the effects of tilting is updated. This is an example of learning action-postconditions.

Another example of analytic learning that is not based on EBL is [Shen, 1989]. Shen gives a system she calls LIVE for learning prediction rules of the form [percepts $\times$ action $\mapsto$ prediction]. The focus of rule learning is to learn the effects of actions based on experimentation. The experimenter tries actions with respect to the agent's current percepts and then observes which of its percepts persisted as well as any new percepts. This is an example of learning effects of actions. It also is an example of a symbolic solution to problems solved by RL (see section 3.4.3). I consider a learning technique similar to Shen's.

3.4.3 Inductive Learning: Reinforcement-Based Learning (RL)

Inductive learning is drawing inductive inferences from observed facts in the environment or from information told to the agent. A recent trend in inductive learning, especially as applied to planning, is reinforcement-based learning. Reinforcement-based learning is a trial-and-error process where, upon executing an action, the applicability of actions to situations is updated based on the reinforcement signal received in response to the action. An action elicited by a stimulus is known as a *respondent* [Hendrey, 1969].[2] This is in contrast to actions controlled by their consequences [Hendrey, 1969], also known as operant conditioning.[3]

A style of reinforcement-based learning popularized by [Sutton, 1984] and followers

[2] Respondent conditioning is also known as instrumental learning. An example of respondent conditioning is laughter as a consequence of being amused.

[3] An example of operant conditioning is laughing at the boss's jokes.

is called Reinforcement Learning (RL). RL considers structural and temporal credit assignment of rewards. I make this distinction so I can talk about other systems that use reinforcement but that are not strictly based on the RL paradigm.

[Maes and Brooks, 1990] is an example of using reinforcement-based learning in planning, different from RL. Maes and Brooks describe a robot that learns to walk. They explored an application of learning to the subsumption architecture of behavior-based AI [Maes, 1990]. Their scheme differs from RL in that while RL only considers positive feedback, their system negative feedback. In [Maes and Brooks, 1990], actions can be executed in parallel. Also, their algorithm works on reinforcing action-preconditions and not state-action pairs as in RL. This work is an example of learning action preconditions.

Another example of reinforcement-based learning that differs from RL is that of [Mani and Uhr, 1992]. Mani presents a system that deliberates about its actions based on a combination of perception and internal needs and then observes the effects of actions, i.e., resulting states. It then strengthens the links between the starting state and the state that results when the action is performed. The reinforcement in this work measures the level to which an action satisfies a need; e.g., eating an apple satisfies hunger 80 percent of the time. Unlike the goal of learning state-action pairs in RL, Mani's goal is to learn action-postconditions, i.e., effects of actions.

The general disadvantages of RL are twofold. First, reinforcements from the environment may mislead RL into learning coincidental effects and not true effects of actions. Second, it is highly computation intensive. This makes it unsuitable for most real-time applications. The closest thing to an application of RL in physical robots appears to be [Lin, 1991] and [Mahadaven and Connel, 1991].

Work described in [Mahadaven and Connel, 1991] is an application of RL in which a physical robot learns to find and push a cardboard carton around. Actions are grouped into the subsumption behaviors *find box*, *push box*, and *unwedge box*, with a given priority ordering among them. RL is applied to each behavior separately. This is one of the earliest reports of physical systems that use RL to learn.

3.5 Automaticity

Two rules are presented in [Shiffrin and Dumais, 1981] which some automatic processes obey. These rules are given below.

1. "Any process that does not use general, non-specific processing resources and does not decrease the general, non-specific processing capacity available for other processes is automatic."

2. "Any process that always utilizes general resources and decreases general processing capacity whenever a given set of external initiating stimuli are presented, regardless of a subject's attempt to ignore or bypass the distraction, is automatic."

These rules don't define automatic processes. Roughly, automatic processes are those that are not controlled. Automatic processes are mostly observed in perceptual mechanisms. However, we are more interested in exploring automaticity in motor skills. Shiffrin and Dumas assert that automatization is a major component of cognitive and motor domains. Thus far, development in GLAIR has focused on applying automatization in the motor domain.

3.6 Action Slips

Action slips are errors in acting. These are counterparts of Freudian slips, where the speaker's intentions differ from what is spoken. Just as Freudian slips are slips of expert speakers, actions slips are slips of agents who are expert at performing the task in which the action slip occurs. In both types of slips, the agent pays so little "mind" to the task that the slips go unnoticed. However, in some cases, these can cause major accidents, such as airplane crashes.

Most slips occur in routine activities. I am interested in an agent's action-generating mechanism that allows slips to occur. With practice, most tasks become routine and automatic. This means that the agent exercises less and less control and monitoring over the execution of tasks. There are a few theories of action by Russian and German scientists around the turn of the century that account for action slips [Dunnette, 1983, Frese and Zapf, 1994]. One relatively new theory is due to Donald Norman [Norman, 1981].

Chapter 4

The Impact of Routines on the Mind

4.1 Introduction

I theorize that an agent engaged in routine activities uses four processes to augment, re-represent, and modify its performance and internal representations. These processes are (1) automaticity, (2) habituation and skill refinement, (3) abstraction-by-chunking, and (4) discovery of new knowledge chunks. My proposed four processes may not be all the processes an agent uses in routine situations. However, they are the strongest processes responsible for the agent's improved performance and augmented know-how. I have implemented a computational model for my theory that manipulates and augments the agent's internal representations according to my theory.

My theory spans activities involving discrete motor skills [Schmidt, 1975] which form small-grained actions, as well as larger granularities of activities such as the scenarios used in the classical planning examples [Fikes and Nilsson, 1971]. The processes in my theory are responsible for the agent's learning but the processes are not machine learning techniques themselves. I will use various machine learning techniques as tools in implementing the processes.

Although my theory borrows concepts from psychological theories of motor skills [Schmidt, 1975], automaticity [Shiffrin and Schneider, 1977], chunking [Miller, 1956], developmental psychology of adaptation [Bjorklund, 1989], theory formation [Shrager, 1990], and discovery [Siegler and Jenkins, 1989], my theory does not try to refine or replace them. Instead, I borrow from related theories in synthesizing my theory.

Shrager (in [Agre and Shrager, 1990, Shrager, 1994]) argues that the evolution of organization of activities of an agent involved in routine activities is best studied as an interaction in the world and not entirely in the agent's head. On the surface, this seems to be an obvious observation. Shrager means that (a) the agent's conception of the environmental factors that affect the agent's activities changes over time, and (b) unbeknownst to the agent, the environmental factors that affect the agent's activities change over time.

The change in the agent's observable behaviors in turn changes the agent's interaction with the world, which in turn affects further observable changes, and so on. For example, in the photocopier experiment in [Agre and Shrager, 1990], Sarah starts working with the photocopier, and after a few pages stops closing the recirculating document handler cover because she sees it as unnecessary. Another example is when signals from the copier prime her actions. She starts to get ready for the next page when she sees the three bright flashes from the copier. She gets ready by positioning her hand on the book even before the copier is done with the page.

On the other hand, changes in the state of the world affect the agent's actions and that leads to the evolution of the agent's activities. For example, it is conceivable that Sarah's copying changes when she reaches the last few pages of the book while her conception of the book as an object being copied remains constant.

I agree with [Agre and Shrager, 1990] on the importance of interaction with the world. However, I don't want to address the evolution of routine activities in general. Instead, I have developed a theory of the impact of routine activities on the agent's mental constructs and do not account for all aspects of the evolution of routines in the world.

Like GLAIR, my theory straddles different classes of processes. My theory draws upon the influence of one level on another: how the contents of one level induce content in another level. GLAIR provides different processing centers with different levels.

As [Shiffrin and Schneider, 1977] distinguish between *control* and *automatic* processes, I make distinctions between processes that go on in parallel versus those that compete for attention and control of the agent's actions in GLAIR levels. What distinguishes control processes from automatic processes is (a) the limitations in the amount of memory space used and (b) the level of control over activities in the process. Control processes are cognitive and require "consciousness". Automatic processes involve mostly motor actions and sensory processing and do not require "consciousness". In GLAIR, I place control processes in the KL and the automatic processes in the PML. In the theory of automaticity developed by Shiffrin and colleagues, "virtually all complex and interesting processes of significant duration involve a mixture of automatic and control processes, often taking place in parallel, with each being able to initiate a process of the same type, or of the other." [Shiffrin and Dumais, 1981, p. 112]. This is similar in GLAIR in that most complex and interesting processing involve both the KL and the PML processes. Also in GLAIR, processes in the KL and the PML take place in parallel. A process in each level is able to start a process in another level. An example of a control process starting an automatic process is stopping a car. The sequence of actions involved in stopping a car can be automatic, but the sequence is started by a decision to stop the car, which is a controlled process. This view is consistent with the way most PML processes are started by the KL decisions to act.

I distinguish two types of routines, perceptual and motor. Perceptual routines are evident in stimulus detection processes, shown in experiments by Shiffrin and Schneider [Shiffrin and Schneider, 1977], that become automatic with practice. Motor routines are patterns of actuator signals that an agent repeats. The length and duration of motor routines are parameters in my theory and are determined for each agent.

My theory identifies internal changes in memory structures of the mind due to routine interactions. These internal changes are both within memory structures and between memory structures. I use the notion of *processes* to describe changes. In this chapter, I present my theory and computational model. I leave details of implemented agents to later chapters. In this chapter, I will start by describing the memory structures and relationships between them in GLAIR. I will then describe major tools involved in the processes. These tools include techniques for maintaining the breakdown of goals and techniques for construction and specialization of PMA. Next, I will describe the processes. I summarize the effects of each process in my theory in terms of (a) effects on the agent's memory structure, i.e., internal representations, (b) enrichment of knowledge and skills, and (c) effect on the agent's overall observable performance.

4.2 Structures

I will present two classes of ontologies and memory structures. This will serve as a foundation for contents of what my proposed processes will modify.

4.2.1 "Conscious" Memory Structures and Their Representations

The entities modeled at the KL are *plans, goals, actions, acts, intentions,* and *beliefs.* These terms are described elsewhere, e.g., [Kumar and Shapiro, 1991], and here the reader can think of them as having their commonsense meanings.

In contrast to "unconscious" memory structures, "conscious" memory structures are useful for communication between the agent and other agents. Here the memory structure consists of explicit nodes representing entities, and propositions about entities. I use SNePS [Shapiro and Rapaport, 1987] for modeling the knowledge representation and reasoning. The SNePS Semantic Network Processing System is a knowledge representation and reasoning system. SNePS is used to facilitate representation and reasoning mind concepts of rational agents capable of natural language communication. Nodes represent concepts. Directed, labeled arcs that connect nodes represent binary relationships between nodes. Nodes are either base nodes or molecular nodes. Base nodes are referred to by name, and no arcs emanate from them. Molecular nodes have arcs emanating from them. The set of outgoing arcs from a molecular node is a cable set. However, SNePS users commonly refer to the set of outgoing arcs from a molecular node as a case frames [Shapiro and Rapaport, 1987]. In addition to the commonly used arcs that give rise to SNePS's traditional case frames, the user is allowed to define new arcs. The user can assert molecular nodes and in effect make new case frames.

I am mainly interested in the acting and planning component of SNePS. Given a goal to achieve, the SNePS acting system, known as SNeRE,[1] will try to find an action to execute

[1] An earlier version of the SNePS acting system is known as SNActor [Kumar et al., 1988]. SNePS rational engine (SNeRE) [Kumar, 1994] is a much-improved version of entities SNActor which unifies acting and inference.

to bring about the desired goal. In so doing, it will search for plans using plan-act and goal-plan propositions to find relevant actions. These propositions are encoded in plan-act and goal-plan case frames. Each case-frame consists of arcs and nodes. For example, the plan-act case frame is a node with two emanating directed arcs labeled `plan` and `act`. A node to which a plan-act case frame is attached is a proposition that the node at the end of the plan arc is the plan (a structure of actions) for performing the act represented by the node at the end of act arc. Similarly, a node to which a goal-plan case frame is attached is a proposition that the node at the end of plan arc is the plan for accomplishing the goal represented by the node at the end of the goal arc. Once SNeRE finds a primitive action for execution, it checks for preconditions. If the preconditions hold, the action is executed and the effects of the action are believed; otherwise, one of the unsatisfied preconditions becomes a goal. This is known as having a sub-goal for SNeRE, and the search for finding an executable action resumes.

4.2.2 "Unconscious" Memory Structures and Their Representation

The entities modeled at the PML are *sensations, actuators, commands, behaviors,* and *tendencies*. *Sensations* are atomic terms used in describing perception. *Actuators* are affecting parts of the agent, e.g., muscles or motors. I assume actuators can be commanded independently. A *command* is an instantaneous, atomic control signal for an actuator. In the case of continuously changing actuator parameters, a command is a control signal for the rate of change of the values for the actuator. A *behavior* is a token that is used for alignment with a concept at the KL. At the PML, a behavior can be considered to be a unit of interaction with the world. Each behavior can correspond to a cluster of possibly simultaneous commands. Behaviors are independent of actuators, and there can be more behaviors than actuators. We will see data structures that associate behaviors with commands. Each behavior can instruct each actuator using only one of the commands for that actuator.[2] A *tendency* is a measure of an agent's preference when there are alternative commands. This preference is computed and maintained based on the agent's prior experiences.

[2] My distinction between acts and behaviors and the levels I choose to place them in GLAIR is consistent with the following quotation from H. Collins. "Acts as I use the term, are what humans do when they intend to do something, whereas behavior is unintended" [Collins, 1990, p. 30].

4.2.2.1 Commands versus Schemas

Previous theories such as [Schmidt, 1975] suggested schema-based, i.e., procedural and subroutine-like encoding of memory structures. Schmidt had empirical reasons to believe that agents have a central and parameterized memory structure. One example is writing signatures. A person produces the same pattern of signature with little regard to size of the signature or with small variations depending on what s/he uses to write. This implies that there is a schema for writing signatures and that it is parameterized to resources and writing surfaces.

In the schema theory, a memory structure is a set of step-by-step programs that can be very rich in control constructs to allow concurrency of movements and the influence of one movement on another. A program starts and stops, and other programs have to compete for resources. This method requires an arbitration policy among programs, which I believe is psychologically invalid since there is no evidence for such an arbitration. Furthermore, with this scheme, parts of a schema may be repeated in other schemas.

I suggest that there is no need for schemas. First, repetition of schema parts is redundant. It is unlikely that an agent needs to encode parts of a program in several places. Second, since the schema need not be communicated to the outside world, there is no need to keep it in a separate module to be retrieved in entirety for communication. A strong reason for encoding a structure of events for a special task is for communication to other agents. But at the PML, there is no need for provision of communication. Thirdly, the need for arbitration among programs is a result of this scheme, and there is no reason agents need to have explicit arbitration at the level of programs.

Therefore, I propose that instead of a schema, the smallest units of actuation are memory structures I call commands. Commands are independently encoded in structures that dictate their activation. Since situations and tendencies surrounding the commands are associated with the commands, the commands are autonomous acting memory structures.[3]

[3]Any procedural program can be converted into a declarative counterpart, and that is an analogy for the proposal of shifting from schemas to command structures.

4.2.2.2 Behaviors Modeled as Feedback Loops

Coupling execution with sensing in a closed loop is a common paradigm in the psychology of action [Adams, 1971]. There are two parts to this paradigm. One is that we form expectations for what is being executed, and we monitor actual outcome with expectations. I see this as purely in the realm of the "conscious" level. On the other hand, rapid sensing and acting forms a closed loop: in a tight loop (iteration)– do something and check the environment. The tight loop implements concurrent sensing and activation of commands. With a nontrivial system of commands, this will have all the characteristics of a feedback loop. These systems are known as reactive systems. [Schmidt, 1975] suggested that reactive systems need not have explicit feedback loops. Instead, the feedback loop is implicit. In some circumstances, autonomous agents act more often than they sense. [Langley et al., 1994] refers to behaviors that require fewer attentional resources as *automatic* behaviors. In contrast to reactive behaviors, they consider these behaviors to be open loop and *automatic*. PMA models these behaviors as well as reactive behaviors. In summary, behaviors modeled in PMA have either implicit feedback loops or they are open-loop.

4.2.2.3 On Rewards

Rewards are used for learning. Most typically, rewards are "goodness values" associated with situations. For a given domain or a task, rewards are determined independently before PMA execution and remain unchanged. For each command transition, an estimate of accumulated rewards is maintained, which is computed using an estimation of accumulations of rewards over time. This value is used in delayed credit assignment.

For an autonomous system, either rewards are precomputed or an external agent in the world provides them for the agent. An interesting approach is that of [Nehmzow, 1992], where a target vector is computed automatically. In Nehmzow's system, a set of sensations are specified as an input and a set of motor actions, i.e., commands, are specified as output of a neural network. After a motor action that does not violate a set of instinct rules is executed for 4 seconds, a target vector corresponding to motor actions is generated. The target vector is a vector with the value of the non-instinct-violating motor action in that and all other elements are set to zero. This is like providing positive reinforcement for maintaining an instinct rule. This idea is very appealing in that I can extend the concept to a structure of needs in intelligent agents. Certain needs can be preservation needs as in instinct rules and other needs can be fulfillment needs. Preservation needs specify maintaining invariant conditions whereas fulfillment needs specify changes in the agent or its world.

4.2.3 Alignment between Terms in GLAIR

When GLAIR levels are connected and data is transfered from one level to another, some objects denoted by terms used in the source level are one-to-one associated with terms used to denote the same object at the target level. I.e., there is a correspondence between terms in the knowledge representation and reasoning system, on the one hand, and sensory perceived objects, properties, events, and states of affairs in the world and motor capabilities, on the other hand. I call this correspondence *alignment*. I will discuss these exact mappings in the context of projects and domains later in chapter 6. Many other objects do not have a one-to-one mapping. For example, a sequence of commands at the PML may map into a single primitive act (see [Kumar et al., 1988, Kumar, 1989, Kumar and Shapiro, 1991] for primitive acts) at the KL. For other objects no mapping exists. For example, intentions do not map to anything at the PM-level.

4.3 Tools and techniques

4.3.1 Perceptuo-Motor Automata

PMA is a system of stimulus-response generation at the PML of GLAIR. In this section, I explain my assumptions about how stimulus-response at the "unconscious" level operates and how we have built a system that models it.

In order to generate commands for actuators, I have developed a representation mechanism and a modeling tool for representing and generating behaviors at the "unconscious" level of an autonomous agent [Hexmoor and Nute, 1992]. PMA is a tool used in modeling PML behavioral components of a GLAIR-agent. I use the word PMA (a) to refer to the system that is used as a tool, as well as (b) to refer to the behavior modules. Therefore, I might refer to a PMA for *chewing* behavior as opposed to a PMA for *walking*. My use of the term will be clear from the context.

PMA as a system is designed to constantly produce signals or impulses for the actuators. I consider these signals or impulses to be instantaneous controls that determine actuator activation levels. For example, for a mobile robot, signals would be activation levels of motors and each signal/impulse will be a fraction of a second in duration. In the context of a GLAIR-agent, PMA models rapid and automatic behaviors of the agent.

Let's assume an agent has a finite number of actuators. As I noted earlier, each behavior can instruct each actuator using only one of the commands for that actuator. Multiple behaviors can be active at the same time. Therefore, an actuator can be commanded simultaneously by multiple behaviors. In PMA as part of a GLAIR-agent, all commands for an actuator issued by different behaviors are passed directly to the lowest level of GLAIR, SAL.

The set of behavioral modules that PMA models in a domain is formally represented by the 7 tuple <Sensations, Commands, Command-Transition, Extended Command Transition, Behaviors, Behavior-Transition, Rewards>. Sensations are atomic (i.e., non-decomposable) terms used in describing perceptions. For example, a sensation for a mobile robot that walks in building hallways can be its *distance to the wall*. I call a pattern of sensations that are used for input to a PMA a situation. A situation which is an n-tuple of Sensations is a PMA input.

Commands and rewards were described in section 5.2. Command transitions (CT) are transformations that produce commands for execution from inputs of situation or situation-behavior pairs (see Figure 4.1). When the latest sensations along with the current behavior match a command transition, that command transition is activated. When a CT is activated, a single command among the command set for the CT is selected for execution.[4]

Given a situation and a behavior, CTs contain a command selection goodness value for each command. This is similar to the weight parameters on the links in neural networks. In my scheme, the goodness value changes with a variant of reinforcement-based learning much the same as update rules change weights on arcs as a neural network is trained. With reinforcement-based learning, goodness values are changed as a function of rewards. This

[4]My implementation of Perceptuo-Motor Automata (PMA) is object-oriented and written in Common Lisp and C. My descriptions of PMA operation therefore consider CTs as objects that can process their input and activate output nodes.

is in contrast to changing weights as a function of differences between activation levels of output nodes and desired output (also known as target values). CTs may take into account an estimated accumulated reward for commands. I call the latter "tendencies". Tendencies are computed using reinforcement-based learning techniques [Sutton, 1988].

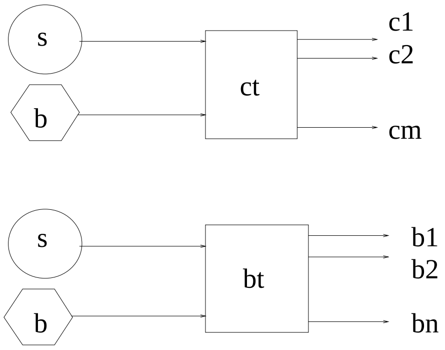

Figure 4.1: A command transition and a behavior transition are shown.

The simplest CTs are mappings of Sensations to Commands. CTs can be disabled by inhibiting them. Inhibition is useful when the agent needs to override "unconscious" behaviors generated by PMA with "conscious" behaviors generated by the KL. The following is all possibilities for Command Transitions (CT) types. The items in brackets represent optional elements of a CT.

[Previous-command $\times$] [Behavior $\times$] Sensations $\mapsto$ command-set [$\times$ Tendency]

For example, let the command set consist of *turn-left* and *turn-right* with tendencies 0.6 and 0.4 respectively; let *Don't-see-anybody* be a sensation at the SAL; and let *Look-for-someone* be a Behavior. Then:

look-for-someone $\times$ *don't-see-anybody* $\mapsto$ *turn-left,0.6* $\times$ *turn-right*,0.4

is a CT. If the sensation and behavior stay constant, 60% of the time *turn-left* will be selected and 40% of the time *turn-right* will be selected.

Now, if we had the following two CTs:

turn-left $\times$ *look-for-someone* $\times$ *don't-see-anybody* $\mapsto$ *turn-left*,0.6 $\times$ *turn-right*,0.4

turn-right $\times$ *look-for-someone* $\times$ *don't-see-anybody* $\mapsto$ *turn-left*,0.4 $\times$ *turn-right*,0.6

the probability of *turn-right* and *turn-left* will be 0.50.

Extended Command Transitions (ECT) are similar to CTs but instead of a set of one or more commands, the consequent is a set of one or more command sequences, each of which may have one or more commands. The significance of ECTs is that once a command sequence starts executing, execution of commands from the sequence may have to compete with commands generated by other CTs or ECTs. Whereas with CTs, arbitration is a local issue, with the addition of ECTs, the arbitration among commands becomes a global problem. I will discuss this arbitration later in this chapter.

Behavior Transitions (BT) are transformations that update the currently active Behavior. When the agent decides on a different course of action at the KL that effects a change in Behavior at the PML, a BT is used to guide the change of behavior. Alternatively, a BT can be used at the PML without a "conscious" decision. A BT is a function: Behavior1 $\times$ Sensations $\mapsto$ Behavior2, where Behavior1 and Behavior2 are Behaviors. For example, *look-for-someone* $\times$ *found-someone* $\mapsto$ *stop* is a behavior transition which selects stopping, after having found someone while the behavior was look for someone.

Behaviors, Behavior-Transitions, and Rewards in a PMA can be nonexistent. The simplest class of PMA is when all three of these elements are empty. Below is the list of combinations of tuples, each defining a class of PMA. For example, <Sensation, Commands, Command-Transitions, NIL, NIL, NIL> models a PMA consisting of sensation/command pairs. It does not include PMAs partitioned into behaviors or rewards in CTs. Therefore it allows the possibility of nondeterminism, i.e., one situation can be mapped into several commands.

- <Sensation, Commands, Command-Transitions, NIL, NIL, NIL>
- <Sensation, Commands, Command-Transitions, Behaviors, NIL, NIL>
- <Sensation, Commands, Command-Transitions, Behaviors, Behavior-transitions, NIL>
- <Sensation, Commands, Command-Transitions, NIL, NIL, Rewards>
- <Sensation, Commands, Command-Transitions, Behaviors, Behavior-transitions, Rewards>

PMA is not all that interesting in itself. The capacity to produce asynchronous and concurrent behaviors, and concurrent commands for actuators allows modeling interactions of the agents in the world. What is interesting is (a) the serendipity of behavior combination unbeknownst to the agent, i.e., emergent behaviors, and (b) how the agent makes use of successful interactions in the world by various learning techniques. PMA is used as the agent's lowest level mechanism to learn from the positive interactions in the world.

In the next section, I describe a data structure for maintaining information about goals and how they can be used to monitor an agent's progress.

4.3.2 Goal Sketches

I propose using the psychological concept of a *goal sketch* [Siegler and Jenkins, 1989]. I will use a goal sketch as a data structure for maintaining constituent attributes of a goal. I have not found a concise definition in psychology for goal sketch. For my use, I define a goal sketch as a knowledge (or data or memory) structure that encodes an agent's understanding of a goal. I will elaborate on this definition later in this section.

In relation to the planning literature in AI, one might be tempted to consider a goal sketch as a subgoal hierarchy. In planning, a subgoal hierarchy is (a) in service of planning, and (b) used as a macroscopic mechanism for plan elaboration and a prescription for action selection. However, I consider a goal sketch to be (a) in service of discovery of new strategies, and (b) a microscopic mechanism for credit (reward) assignment. In addition, a subgoal hierarchy as used in a goal sketch need not be a complete decomposition of the goal as typically used in planning.

Goal sketches were first proposed in connection with children's learning of new strategies [Siegler and Jenkins, 1989]. In an example, it was illustrated that the min strategy for addition[5] is discovered by a goal sketch for addition.

I note that, having a goal to perform a task, children can observe other people engaged in an interaction for the task. What they observe will be a new strategy which is consistent with their goal sketch for the task. Their observation of a novel strategy is a new discovery. This is similar to when children accidentally manage to accomplish a difficult task or manage to accomplish a goal by a new strategy.

Using a goal sketch, an agent can determine if a strategy brings about a goal. Goal sketches are not stand-alone knowledge structures. A recent study reported in [Siegler and Crowley, 1994] indicates that (1) an agent can tell how well a strategy fits a goal

[5]With two numbers to add, the min strategy is to count up from the larger number by the smaller number.

independently of whether the agent uses that strategy for the goal, (2) a goal sketch is not a self-contained strategy mechanism, but it serves as a constraint on a strategy discovery process.

Regardless of whether an agent already has an effective plan to accomplish a goal, the agent may decide to discover a new effective plan (i.e., a new strategy) for the goal. I call a goal for which an agent has a mental attitude to discover a new strategy "a goal for discovery." A goal for discovery differs from a goal to be accomplished in that, since a new effective plan for the goal for discovery is needed, the agent does not immediately accomplish that goal. A goal for discovery can be decomposed into other goals.

In SNeRE, a goal exists only in goal-plan case-frames. Goal-plan case frames specify that a plan is a way of achieving a goal. As a goal-plan case frame is used in planning, given a goal, a plan can be retrieved. For achieving a goal, the agent looks for a plan associated with the goal. While searching through a plan for executable actions, the agent may discover subgoals, which will make the agent look for plans to accomplish these subgoals, and so on. Unlike that scheme, in working with a goal sketch, the agent tries to achieve all subgoals concurrently and independently. Also, with a goal sketch, there is no need to search for plans. The agent is not actively trying to achieve the topmost goal for discovery in the goal sketch by looking for a plan. In this respect, the goal for discovery is more of a desire to find a strategy than it is a goal for accomplishment.

Next, I will outline my goal sketch approach, its location in GLAIR, and goal sketch properties. I make a distinction between an agent's automatic interactions with the environment and control interactions, as shown in Figure 4.2. This figure shows at a high level that a goal sketch is used in learning two types of interactions in the environment. Goal Sketch Manager (GSM) is a program which interprets a goal sketch. GSM matches external stimuli and the agent's stimulus/response patterns against parts of a goal sketch to reinforce and discover patterns of interaction.

GSM performs two things: (1) keeps track of a goal sketch progression by issuing rewards and punishments to be used in the PML, (2) when a structure is discovered at the PML that satisfies a goal sketch, GSM attaches that structure in the KL as the effective plan for the goal sketch.

Generally, a goal sketch has to sufficiently specify the start condition of a scenario so

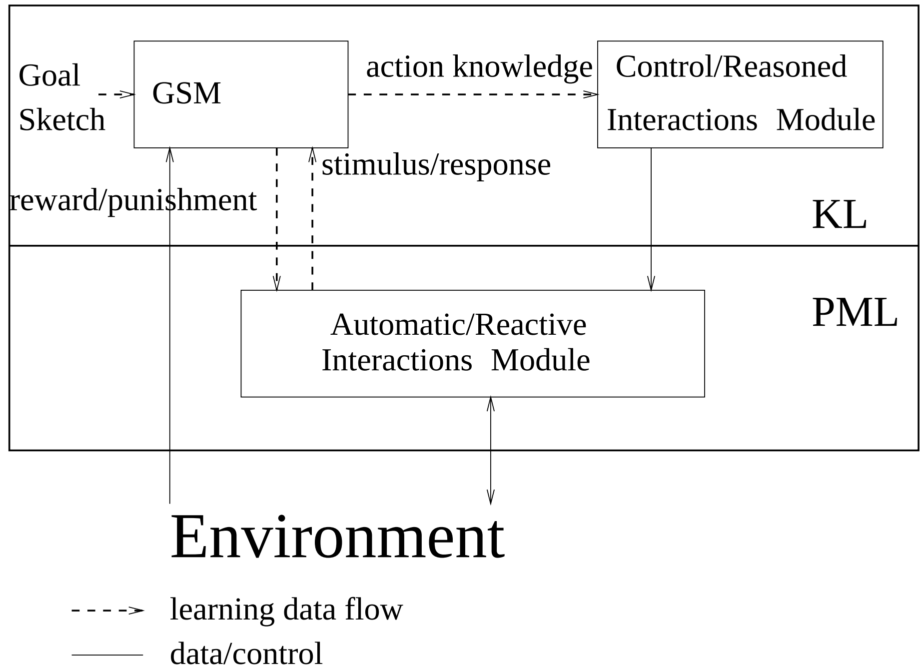

Figure 4.2: Schematic of role of goal sketch in my approach: GSM uses situations in the environment to compute reward and punishment for acting modules in the PML, and it makes new connections between disparate knowledge chunks at the KL.

the agent is primed to "attend to" the sequence of events that follow. A goal sketch must also sufficiently specify the end condition so the agent will know "when to stop paying attention". In the foregoing discussion, I use the notion of attention loosely in so far as having a short-term memory for events.

Recent work in machine learning has shown how a variety of reinforcement-based learning techniques can be used to determine the appropriateness of actions in given sensory circumstances using rewards [Lin, 1992]. A goal sketch can be used to structure a reward system that can be used in conjunction with a reinforcement system of learning. However, the major goal of this section is to show that various interactions in the world can be enhanced with the use of a goal sketch.

I make an assumption that a goal sketch can be used to produce rewards and punishments for automatic behaviors. This is an assumption that is not yet psychologically validated. To avoid confusion with psychologically validated theories, I characterize a goal sketch as having the following properties:

- A goal sketch specifies the start, one or more milestones or conditions that signify a measure of progress or satisfaction, and the end of a goal-directed activity. For example, a project-plan (which is a project management tool and not a plan in the AI sense)[6] can be considered to be a goal sketch.

- A goal sketch is not a plan in that it does not prescribe a pattern of actions to be performed. A goal sketch also differs from a plan in that it is more abstract.

- A goal sketch does not consume all of the attentional capacity of the agent. Thus, an agent can monitor a goal sketch and attend to other activities as well.

- Making positive progress along a goal sketch is considered to yield rewards, and making negative progress on a goal sketch is considered to incur punishment. Rewards and punishments from a goal sketch are atemporal and are not a function of agents, actions, or reasoning. Rewards and punishments are a function of the stimuli and perceived state of progress and satisfaction of a goal sketch.

4.3.2.1 Formalizing the Goal Sketch

A goal sketch is represented as a goal hierarchy (tree) structure [Siegler and Jenkins, 1989]. In my characterization of a goal-sketch, **goal sketch is a hierarchical decomposition of a "goal for discovery" into other goals which might be other "goals for discovery."** The leaf node goals of the hierarchy tree are propositions about the smallest perceptual units that are pertinent to a goal sketch. One or more leaf nodes will be propositions about conditions that signify satisfaction of the goal sketch. I call these the termination nodes. Similarly, one or more leaf nodes will be propositions about conditions that signify the start of progress in the goal sketch. I call these the start nodes. A goal hierarchy is activated when one of its start nodes is activated. A leaf node is activated when the proposition at the node holds. Given any start node, I assume that progress toward any goal node identifies an ordered list of leaf nodes. This implies that the goal-sketch traversal algorithm is fixed. In SNePS, I will represent propositions about the goal hierarchy using

[6] An AI plan is an effective strategy for accomplishing a task, and it suggests steps for execution; whereas a project-plan is a specification scheme for monitoring progress in a project, and it does not suggest steps for execution.

goal-subgoal and goal-node case frames. All nodes that has no goal arc pointing to them are goal sketch leaf nodes. With each leaf node, we will have a rule that assigns a reward value for satisfaction of the node. Later in this chapter we will describe an algorithm using SNePS structures for using goal sketches to issue rewards as well as to discover new strategies.

4.4 Processes

4.4.1 Process of Automaticity

Automaticity is a process by which the agent's performance of perceptual and motor activities become faster. In my computational theory, the process of automaticity is responsible for learning PMA transitions at the "unconscious" level or lessening of "conscious" deliberation of the knowledge with PMA practice.

In GLAIR, automaticity is illustrated by augmented command transitions. Consider primitive actions a, b, c to be part of a complex action A. A can be decomposed into a choice between a, b, and c. Suppose we have no command transitions at the PML that can be equivalent to these actions. Under "conscious" control, whenever action A is being executed in situation S1, suppose one of a, b, or c is chosen for execution. Say a is chosen. At the PML, a command transition $\mathrm{S1} \times \mathrm{B} \mapsto \mathrm{a}'$ will be built. B in the command transition is considered to be a behavior corresponding to action A and the command a' corresponds to action a. Similarly, command transitions $\mathrm{S2} \times \mathrm{B} \mapsto \mathrm{b}'$, $\mathrm{S3} \times \mathrm{B} \mapsto \mathrm{c}'$ are built when A is decomposed into b and c under situations S2 and S3 respectively. S1, S2, and S3 may be the same situation. Once command transitions corresponding to A are built, the agent has built "unconscious know-how" that can potentially replace the use of "conscious" knowledge.

I will discuss the circumstances for preferring to use "unconscious know-how" in section 4.4.3. I will also discuss how complex acts at the "conscious" level may be treated as primitive acts.

The effects on the agent's memory structure are that the agent develops a repository of reasoned out stimulus-response pairs at the "unconscious" level. The agent will have new skills that previously corresponded with a cognitive task. One observable effect of this process is that the cognitive task of planning becomes faster and is substituted with sub-cognitive activities. The result is faster and more continuous interaction in the world.

4.4.2 Process of Habituation and Skill Refinement

The process of habituation and skill refinement is a process by which the agent settles on a set of unconscious preferences, or improves its performance by finding unconscious successful preferences, or both. In GLAIR, the process of habituation and skill refinement is entirely in the PML. The process makes use of environmental rewards as well as internal conditioning of habituation to enable an agent to use successful and familiar courses of action.

Habituation can be considered to be a form of internal reward system. The agent rewards itself (i.e., feels good about) repetitious courses of action. Habituation is commonsensically understood as a repetition of a command under similar situations. However, [Dunlap, 1932, p. 9] states that "Any habit whatsoever is really a tendency to a response which varies constantly within certain limits, and the 'invariability' of a habit is determined by the limits of measurement employed in describing the habit." For my present purposes, I assume we can measure habituation with repetition and avoid issues of variability. I define a habit as "doing the same action at the same time or repeating a technique in the same way under the same situations." So, either an action is identically repeated at fixed intervals of time, or the technique is identically carried out in a given situation. Habits, with this simple definition, can be very useful in learning from routine activities. "At the same time" in my definition refers to time of day, week, month, or year. Here, the time of day is the situation that prompts the agent to do something. For a non-human agent, time can be divided into pieces that are like days.

Many times, simple habituation by repetition prejudices further progress. "Having mastered a clumsy stroke, which makes it possible to keep afloat, the learner uses this stroke for a period of time, not attempting to improve it, but merely to make it habitual. If he stays at this stage too long, he certainly prejudices his further progress" [Dunlap, 1932, p. 231].

This process does not always lead the agent to the optimum path. However, the agent learns to repeat familiar sequences of action, and, as a result, the agent may interact more deterministically in a given environment. By deterministic interaction, I mean making the same choices in the given situations.

My assumption for the process of habituation is that we have an agent who has some basic skills in the form of action transitions and behavior transitions, but for many situations she can't choose between alternatives because she doesn't know the consequences of her actions.

I will present two techniques in GLAIR for this process. The first is an extension of reinforcement-based learning and the other is using goal sketches. Whereas my reinforcement-based learning technique helps with habituation, my goal-sketch technique aids in skill refinement. These techniques are independent of one another and can be used simultaneously.

4.4.2.1 Extending Reinforcement-Based Learning

Two components of the process of habituation and the reward structure using reinforcement-based learning are seeking "desirable" situations and repeating "what worked before". The first component requires that the agent's interactions are purposeful and rewards are given based on how well the agent reaches its goal or completes its task. The second component requires that the agent inherently seeks to use its experience from prior interactions. The rewards in the second component are derived from the frequency of interactions.

I propose two learning techniques which in turn use command transitions and extended command transitions. The first is a modified Q learning [Watkins, 1989]. In Q learning, **Utility**(S(t),C(t)) is the evaluation of how appropriate command C is in situation S when C is executed at time t in response to S at t. **R**(S(t+1)) is the "reward" received by being in state S(t+1). The following equality is maintained and propagated as each command is executed:

$$\begin{aligned}\textbf{Utility}(\mathrm{S(t),\ C(t)}) \quad = \quad & \textbf{R}(\mathrm{S(t+1)}) \\ & + \gamma\ [\lambda\ \textbf{Utility}(\mathrm{S(t+1),\ C(t+1)}) \\ & + (1 - \lambda)\ \max_{C(t+1)} \textbf{Utility}(\mathrm{S(t+1),\ C(t+1)})]\end{aligned}$$

where γ is a parameter to determine how important it is to be in the state that the agent ends up in after his move. In reinforcement-based learning, this is known as the discount factor, and λ is known as the recency or learning factor. My modified update equation takes the previous command into account:

$$
\begin{aligned}
\mathbf{Utility}(\mathrm{S(t), C(t), C(t\text{-}1)}) \quad = \quad & \mathbf{R}(\mathrm{S(t+1)}) + \mathbf{RI} + \\
& \gamma\ [\lambda\ \mathbf{Utility}(\mathrm{S(t+1), C(t+1), p(t+1)}) + \\
& (1 - \lambda)\ \max_{C(t+1)}\mathbf{Utility}(\mathrm{S(t+1), C(t+1), C(t)})]
\end{aligned}
$$

RI is the internal reward and refers to rewards an agent gives itself. This is in contrast to **R**, which are rewards an agent receives from the environment. **RI** can be considered to be subjective, whereas **R** is objective. **RI** can be a function of repetition of commands, i.e., function of frequencies of C(t) by C(t+1). This means I will keep track of how often a command is preceded by another command. I then use **RI** to reward selection of command that is followed most frequently. I consider **RI** as a measure of habituation. I believe **RI** along with **R** will speed up reinforcement-based learning.[7]

My second proposal explains the construction and invocation of the use of extended command transitions. Under certain conditions, repetitions of a sequence of commands become so automatic that the agent will act as if she is entirely mindless of the situations prompting the commands. Normally, an agent is in a continuous feedback loop of checking a situation and getting a command transition to determine the most appropriate command for execution. However, once an extended command transition has produced a command sequence, the agent has a tendency to go through the sequence without regard for checking the situation to find the most appropriate command. For example, tasks that are learned at initial stages are carried out quickly, but still step by step. Once a task is mastered, like changing a flat or airplane takeoff procedures, there is a chance that the agent will not wait for completion of a step before starting on the next step.[8] I consider the command sequence as a routine. In order to build extended command transitions, a sequence C1 ... Cn is observed to be automatic under the following two conditions.

1. The situation/behavior pairs S_0/B_0 ... S_{n-1}/B_{n-1} which correspond to commands C_1 ... C_n being selected for execution are repeated for a number of times.

2. Commands C_1 ... C_n are repeated for a number of times.

[7]If **RI** were a function of frequency of repetition of S(t) by S(t+1), a situation that precedes another situation, **RI** would reflect a measure of structural credit assignment[Lin, 1992]. Structural credit assignment is the problem of how to propagate the reward spatially across states so that similar states cause the agent to take similar actions.

[8]This has been the source of many accidents.

The actual number of repetitions needed is a matter of psychological testing. In my computational model, I use a threshold for this value.

At each cycle of PMA, each CT uses the goodness values and locally decides on a command for execution. For the sake of promoting learning new parts of the problem space and effects of commands, 90% of the time the agent will choose the command with the highest goodness value.[9] The remainder of times, the agent randomly picks among the candidate commands. With the addition of ECTs, once a command sequence is being executed, if the behavior that triggered the ECT changes, execution of the sequence is terminated. While a sequence is being executed, the PMA continues checking the situations and fires CTs or ECTs whenever they are applicable. If the behavior remains in effect through execution of a command sequence and if a situation leads to generation of a command that is different from the one in the sequence; 90% of the time I opt for ignoring the command that is different than the one in the sequence and continue with the command in the sequence. The other 10% of the time, I choose the new command so the problem space can be further explored. Should the agent not exercise the command sequence routine and follow what seems appropriate? To answer this question, psychological tests are needed but they are beyond the scope of my developing a computational model.

4.4.2.2 Using Goal Sketches

My second proposal explains a form of habituation. This form of habituation seems useful when the agent does not have adequate rewards from the outside.

The GSM examines the match between a command and behavior transition and leaf-node activation to generate a reward or a punishment. Which parameter or what transition uses the reward and punishment values is not known in the goal sketch.

When a leaf node is activated, either a reward or a punishment is issued by the GSM. If the current situation matches a leaf node, the goal at the leaf node is satisfied. The reward rule associated with the node is forward chained and a reward is issued. If the current situation matches negation of a leaf node, the negation of the goal at the leaf node is satisfied. The punishment rule associated with the node is forward chained and a

[9] This percentage is an experimental percentage; it has been shown to be a good value in systems that learn their environment [Lin, 1992].

punishment is issued.

Given a behavior B and a situation S, a command transition CT and a behavior transition BT is identified. Prior to learning, I let all commands for CT have an equal chance of being selected for execution. Similarly, all consequent behaviors for BT have an equal chance of being selected. In other words, I associate an equal goodness with each command in CT and behavior in BT. If, after execution of command c_i, which is selected arbitrarily for CT, a reward is issued by the GSM, then the goodness of c_i is incremented:

goodness(CT, c_i, t) = goodness(CT, c_i, t–1) + gs-reward(t)

This makes c_i the most preferred command in CT. I treat the goodnesses as probability distributions. This makes the choice of a command for CT a random variable. Rewards received over time change the odds for choosing commands. Punishment of one command in CT is treated as a reward for the competing commands and increments their goodness. The equation above can easily be extended to account for discounted cumulative rewards as in Q-learning [Lin, 1992]. I have done this in a few of my GLAIR agents.

Similar to command transitions, if a behavior transition is executed and the choice of new behavior results in a reward, its goodness is updated according to the following equation:

goodness(BT, b_i, t) = goodness(BT, b_i, t–1) + gs-reward(t)

If a behavior transition leads to a punishment, the competing behaviors are rewarded. Since deciding what to do next is delegated to PML, this process eases the agent's mental burden.

In redundant environments, actions become more deterministic. Executing actions requires less feedback from the environment, and movements become contiguous. With this process, the agent will have established a preference for follow-up commands given a command just executed and the subsequent situation just observed. The agent gains a preference for commands and command sequences. Since this preference is in the PML, it is a form of skill acquisition. The effect of this process is entirely observable. Since the agent gains preferences for commands, the cognitive burden is lowered. The agent can achieve increased speed and continuity in sequences of command executions.

4.4.3 Process of Abstraction: Chunking

The process of chunking is a process by which the agent treats a collection of memory structures as a singular structure. As a result, the agent becomes capable of handling more complex perceptual and motor activities. In GLAIR, as an agent learns to interact more at the PML, actions at the KL that used to be complex acts may become primitive. I consider this as a form of *chunking*. The concept of chunking was introduced in [Miller, 1956] as units that are formed when a human subject receives sensory input. The chunks were used to explain how people learn to recognize larger and larger pieces of input despite a fixed immediate memory span. By forming ever larger chunks, the subject is thought to be *recoding* the input into abstract, readily identifiable chunks. In GLAIR, the conditions under which this process leads to chunking are: (a) the agent learns a sequence of actions in the "unconscious" that used to be available only "consciously", (b) the agent learns certain stimulus–response pairs in the "unconscious".

Consider a complex action A which can be decomposed into a sequence of primitive actions a, b, and c. At the PML, let's have command transitions S1 $\times$ a$'$ $\mapsto$ c1, S2 $\times$ b$'$ $\mapsto$ c2, S3 $\times$ c$'$ $\mapsto$ c3 where S1, S2, and S3 are three situations. a$'$, b$'$, and c$'$ are behaviors at the PML that correspond to actions a, b, and c at the KL. c1, c2, and c3 are commands with no corresponding actions at the KL. Under "conscious" control, whenever action A is chosen, a is called, which makes command c1 be executed provided S1 holds. After a is executed, b is called, which will make c2 be executed if S2 holds. S1 $\times$ $a \mapsto b$ is learned as a behavior transition if it does not already exist. Similarly, after c is called, command c3 is executed (if S3 holds), and (S2 $\times$ b) $\mapsto$ c is learned as a behavior transition. The two behavior transitions that are possibly learned at the "unconscious" contain the sequentiality of actions a, b, c. Having learned the behavior transitions, the agent can, and will, use the behavior transitions, under most circumstances, over the decomposition of action A. By using the behavior transitions, instead of elaboration of A, the agent learns to treat A as a primitive action and no longer elaborate it.

My use of chunking is similar to Soar's [Laird et al., 1987] in that new rules are introduced to cache part of reasoning and problem solving. Chunking in SOAR is applied to production rules to generate new production rules which shortcut chains of reasoning. I apply chunking to my representations at an "unconscious" level and the result is marking complex acts as primitive at the "conscious" level. Similar to SOAR, this is a shortcut in reasoning– in our case a shortcut in plan elaboration. Unlike SOAR, our chunking operates between two levels with different representations.

The effects on the agent's memory structure are that the agent will have developed a new organization of knowledge. Actions previously classified as complex become primitive. The agent will have new skills that previously corresponded with a cognitive task. The agent's knowledge is abstracted. Observable effects of this process are that the cognitive task of planning becomes faster and is switched earlier to sub-cognitive activities. The result is faster and continuous interaction in the world.

4.4.4 The process of discovering new knowledge chunks

The process of discovering new knowledge chunks is a process by which the agent induces knowledge using automatic self-observations of implicit know-how. In my computational theory, I propose discovery of action preconditions, effects of actions, and action sequences at the KL by observing implicit know-how at the PML. The process of discovering new knowledge chunks is used when an agent suddenly discovers new knowledge or a new relationship among disparate pieces of knowledge.

I consider the additions at the KL to be by definition *emergent* [Smith and Stevens, 1994]. Everyday examples of this process are (a) Archimedes's finding out the weight of gold; when he discovered that he can determine the weight of king's golden crown by water displacement, he shouted "Eureka", (b) when young children discover how things work like the relationship between shapes of glasses and their capacity to hold liquids, and (c) when people expert at tasks have mishaps, they learn consequences they might not have known. Instead of scientific discoveries, I am interested in everyday discoveries. It is not clear what leads to these everyday discoveries. One account of this process is that new relationships are being developed gradually, and only when the strength of the relationship reaches a certain level, the agent becomes explicitly aware of the relationship. Another account of this process is that of an instantaneous realization of a relationship. This latter account seems to fit the "Eureka" effect. In the examples I sighted above, the agent seems primed for finding an answer to a question. Although the agent may not be spending all its resources in making a discovery, it is aware of relevant occurrences in the environment.

The gradual building of a relationship account seems favored for explaining children's discoveries. Children tend to try many ways of performing a task and in the process gain know-how, for a broad range of interactions. Despite their varied know-how they often can't piece together new strategies or recognize new strategies. However, they remember their well-practiced strategies. Following [Siegler and Jenkins, 1989], we hypothesize that children maintain a goal sketch which specifies what a successful strategy must satisfy.

Occurrence of an *action slip* make the agent aware of the "unconscious" components such as a plan, a policy, or preferences in acting. The agent examines its "conscious" intentions and how the "unconscious" know-how fits its goal sketch at the KL. I hypothesize that either the "unconscious" know-how is learned at the KL or the combination of the corrective actions forms the basis of a new strategy. This is an example of an immediate, non-rehearsed account of discovery.

Using goal sketch, the GSM matches KL patterns which results in new constructs in the KL. Learning at the KL using a goal sketch is an aftermath of PML learning. This is because the agent makes realizations based on its own interactions. Matching for KL takes account of both KL elements and PML constructs.

4.4.4.1 Preconditions, Effects, and Action Sequences

Let's use the model that primitive actions in KL map to behaviors in the PML. Let's consider successive steps in time: t1 and t2. Let G be the leaf node of currently active goal sketch at the KL and it is satisfied at t2. Let S be the current situation at t1 (just before G was satisfied). S is in the antecedent of the CT executed at t1 in the PML. Let P be the proposition at the KL corresponding to S. Since S holds in PML, I say P is believed at the KL. Let S $\times$ B1 -> C be the CT and S $\times$ B1 -> B2 be the behavior transition BT, which are executed simultaneously at t1. Let A1 be the primitive action corresponding to the behavior B1 and A2 be the primitive action corresponding to the behavior B2.

I suggest we can conclude some rules to be learned at the KL, provided: (a) the above pattern occurs with high frequency, (b) B1 and B2 are not the most frequently occurring behaviors, and (c) G is not a common goal. I suggest that the agent can conclude that *G is an effect of A1*. This rule is similar to a proposition about effects of an action. I suggest that the agent can conclude that *P precedes A2*. This proposition is similar to a proposition about precondition of an action. I will consider these rules as general SNePS assertion rules and refrain from comparing them to traditional precondition and effect propositions. I suggest that the agent can conclude that *A1 precedes A2*. Obviously, these rules are not provably correct. More realistically, these rules can be considered to be conceived and hypothesized for learning. Since I am working with synthetic agents, I set arbitrary thresholds for the number of repetitions before the agent learns these rules.

Beyond learning short sequences of actions, any pattern of interaction in the PML that satisfies a goal sketch, however crude and redundant the pattern, can be considered to have a corresponding strategy (or a plan) for accomplishing the goal sketch in the KL.

The effects of process of discovery on the agent's memory structure are that the agent will have new relationships between previously unrelated knowledge pieces. This is particularly useful in building new relations among previously unrelated actions. Subsequent cognitive tasks for planning can take advantage of the new relationships. The observable effects of the discovery process are that the agent can plan and, in general, reason with a richer knowledge base about its activities in the world. I.e., the agent has learned more about the effects of its actions in the world and thereby has learned more about the world.

4.5 Concluding Remarks

My theory proposes several processes that take place as agents interact with the world in routine activities. These processes may seem similar to previous psychological or learning theories. To my knowledge, this effort is the first to point out these processes in light of a theory of how the agent's mental and physical interactions change over time.

Routine interactions affect an agent by improving its effectiveness of interactions, faster interaction, lessening cognitive load, and increasing its knowledge of the world.

Chapter 5

GLAIR Agents

In this chapter, I introduce several testbeds and GLAIR-agents. My description will familiarize the reader with (1) the testbed hardware and software, (2) GLAIR components of agents, (3) the agent interactions in the world, and (4) what the agent will demonstrate. How these agents demonstrate parts of my theory about routines will be left for a later chapter.

5.1 Gabby and ABS: Air Battle Simulation—An Experiment in Automaticity, Habituation, Skill Refinement, and Discovering New Knowledge Chunks

With the help of Guido Caicedo and Frank Bidwell, I have written a program, Air Battle Simulation (ABS), that simulates World War I style airplane dog-fights. ABS is an interactive video-game where an opponent (a human player or a fixed strategy computer agent) plays against a GLAIR-agent.

Along the lines of Agre's improvised actions [Agre and Chapman, 1987] and Brooks's subsumption [Brooks, 1985], I believe complex behaviors emerge from interaction of the agent with its environment without planning. However, previous work in this area hard-coded a lot of primitive actions and did not attempt to learn the improvised behavior. I start with a minimal number of primitive actions and sensations. The basis for this minimality and choice of primitive actions is physiological. In other words, in modeling a computer agent, I will choose actions that are physically basic for the robot as primitive actions for the computer agent. I then instruct the robot to perform tasks, and, in the midst of robot accomplishing this, I expect the robot to notice basic behaviors (i.e., routines) emerge. An example of an emergent behavior I will explore is: *when chasing the opponent, learn its evasive maneuvers and take steps to counteract them.* Emergent behaviors will be learned and consequently used in the agent's repertoire of actions.

Figure 5.1 schematically presents the structure of the GLAIR agent (the computer agent) that plays the Air Battle Simulation video-game. I will refer to this agent as "Gabby," for "GLAIR air battler". Initially, Gabby has not acquired a PMA, and so uses conscious level reasoning, i.e., SNePS behavioral rules [Kumar and Shapiro, 1991], to decide what move to make. Once transitions are constructed and cached in a PMA, Gabby uses the PMA for deciding its next move whenever possible. I consider the addition of transitions to PMA as a form of learning. This will be discussed in chapter 6. In addition to adding transitions, Gabby uses other forms of learning that operate on PMA transitions and improve Gabby's performance. Gabby demonstrates real time behaviors and the inter-level alignment mechanism in GLAIR.

The ABS video game runs on a SparcStation. The game display consists of a main window giving a vertical view, a side window giving a horizontal view, two "damage reports" (see Figure 5.2), and a control panel window (see Figure 5.3). The opponent player's plane is always considered to be in the center of the two shaded areas. The horizontal two-dimensional position and orientation of Gabby's plane is displayed by an airplane icon on the main window, and its height relative to the opponent's plane is indicated by the drawing of a plane in the side window. The condition of the opponent's plane is indicated by the report labeled "Own Damage," and the condition of Gabby's plane by the report labeled "Enemy's Damage." When the two planes are close in all three dimensions, as indicated by Gabby's plane being shown in the two shaded areas, whichever plane is facing the other one automatically fires. (That is, neither Gabby nor the opponent makes a separate decision about when to fire.)

The opponent player uses the control panel to choose a move, which is a combination of

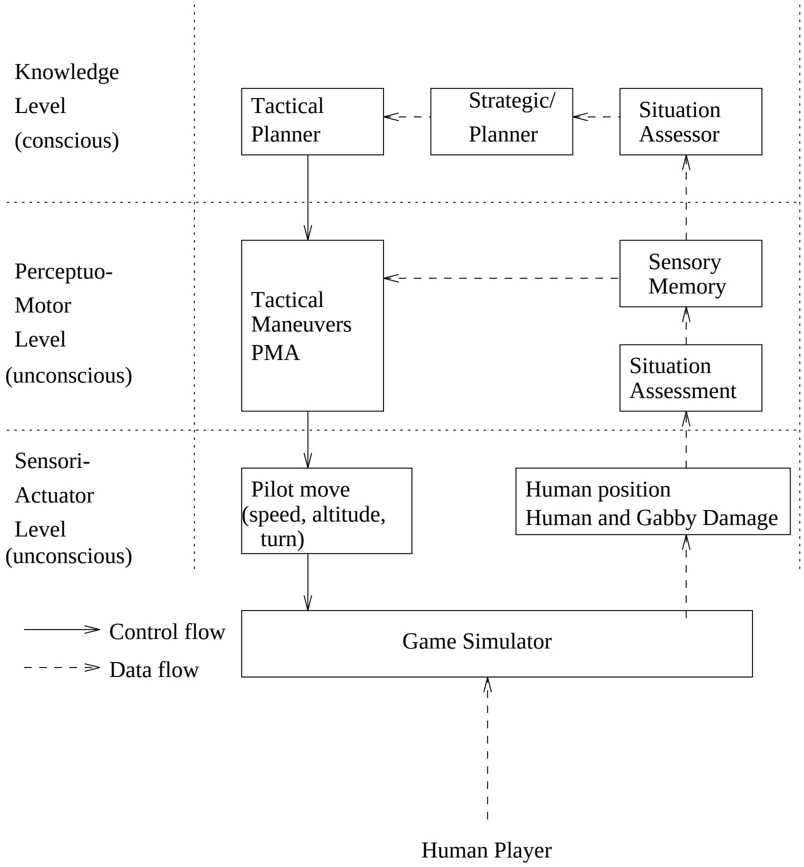

Figure 5.1: Schematic representation of the Air Battle Simulation playing GLAIR agent, Gabby.

changing altitude, speed, and direction. When the opponent player presses the GO button, Gabby also selects a move. The game simulator then considers the two moves to determine the outcome, and updates the screen and the accumulated damage to the two planes, thus simulating simultaneous moves. The game ends when one or both of the two player's planes are destroyed.

5.2 Garry and Gerry—An Experiment in Habituation, Skill Refinement, and Discovering New Knowledge Chunks

Gerry is a 2.5′ tall mobile robot and has two independently driven wheels, an infrared transmitter, four infrared sensors, a front and a rear bumper, and four corner whiskers

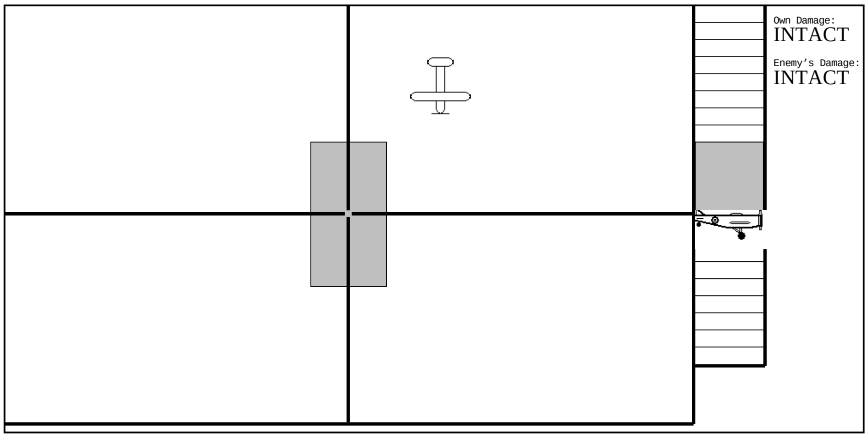

Figure 5.2: Air Battle Simulation game windows. Gabby's plane is indicated by an airplane icon in the upper right quadrant of the main window, and by the drawing of a plane in the side window. This figure shows Gabby heading toward its opponent, and at a lower altitude than the opponent. The shaded regions denote a 3D shooting range. If Gabby's plane appears in both shaded regions, whichever plane is facing the other (possibly both) will fire.

controlled by a 6.270 board.[1] Gerry has four primitive abilities: going straight, turning right, turning left, and stopping. When a whisker or a bumper senses contact with an external object, Gerry performs a reflex. There are reflexes corresponding to each whisker and bumper. Reflexes are implemented in Gerry's SAL. At Gerry's PML, behaviors are implemented using PMA. Currently, Gerry has PMAs and some basic knowledge at the KL for searching for another robot, Garry.

Garry is a GLAIR based 1′ tall Lego robot which is partly controlled by a 6.270 board. Using an RS232 (serial link) and a phone line extension cord, Garry is connected to a Sparcstation. Garry's SAL and PML components are on the 6.270 board, whereas the KL component runs on the Sparcstation. Garry has two independently driven wheels and two infrared sensors. The infrared sensors do not provide distance information. Garry has a front bumper, a rear bumper, and two side bumpers/whiskers. Front and side bumpers are adequate to detect walls when Garry is in motion. Underneath Garry, there are three reflectance sensors in a triangle formation. Garry can tell a black surface from a light/reflective surface.

[1] 6.270 boards are robot controller boards developed at MIT for use in their 6.270 class offering. 6.270 boards have a 6811 microprocessor with multitasking capabilities.

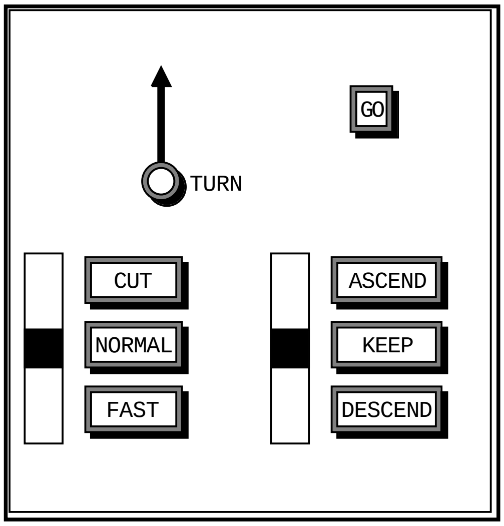

Figure 5.3: Air Battle Instrument Panel window. To select/change a move, the opponent player pushes one of the buttons in each column and adjusts the TURN dial by repeatedly clicking on it. CUT, NORMAL, and FAST are used for speed selection. CUT is the slowest speed selection. NORMAL is the normal cruising speed. FAST is the fastest speed. ASCEND, KEEP, DESCEND are used for altitude change. ASCEND indicates an elevation increase, KEEP maintains the elevation, and DESCEND indicates an elevation decrease. Pushing the GO button submits the move to the game simulator.

Next, I describe two experiments with Gerry and Garry.

5.2.1 Experiment 1: Gerry Chasing Garry

In this experiment Gerry is given the task of chasing Garry and Garry is given the task of evading Gerry. We focus on Gerry as it tries to learn from chasing Garry. Gerry and Garry start at random orientations. Due to limited infrared range, the starting distance between them are less than 2 feet.

Figure 5.4 shows that Gerry's KL knowledge consists of three separate pieces of knowledge for spotting Garry, catching up with Garry, and grabbing Garry. At the KL, Gerry doesn't know that these are parts of a task for getting Garry. Perceptual reduction, which is processing and filtering perceptual data, is implemented by having concepts at the KL correspond to patterns of stimulus at the PML. For example, "no-garry" at the KL corresponds to the pattern of no infrared being sensed at the PML. At the KL, there are no concepts for specific infrared signals that are in the PML. Each action in the KL is expanded into a behavior in the PML. This is an implementation of action elaboration where actions at the PML are more elaborated as compared to the KL actions. As shown in Figure 5.4, action elaboration is implemented by mapping an action in KL to a behavior and corresponding command transitions. For example, "spot-garry" is shown to map to a behavior spot-garry in PML and the solid box at the PML shows six command transitions. Command transitions in the PMA map infrared situations to commands. By ways of abbreviation, I have left out behaviors from command transitions. In Gerry, each situation in the PMA is mapped to all commands. What determines the applicability of a command given a situation is their goodness level reflected by the gi in the Figure.

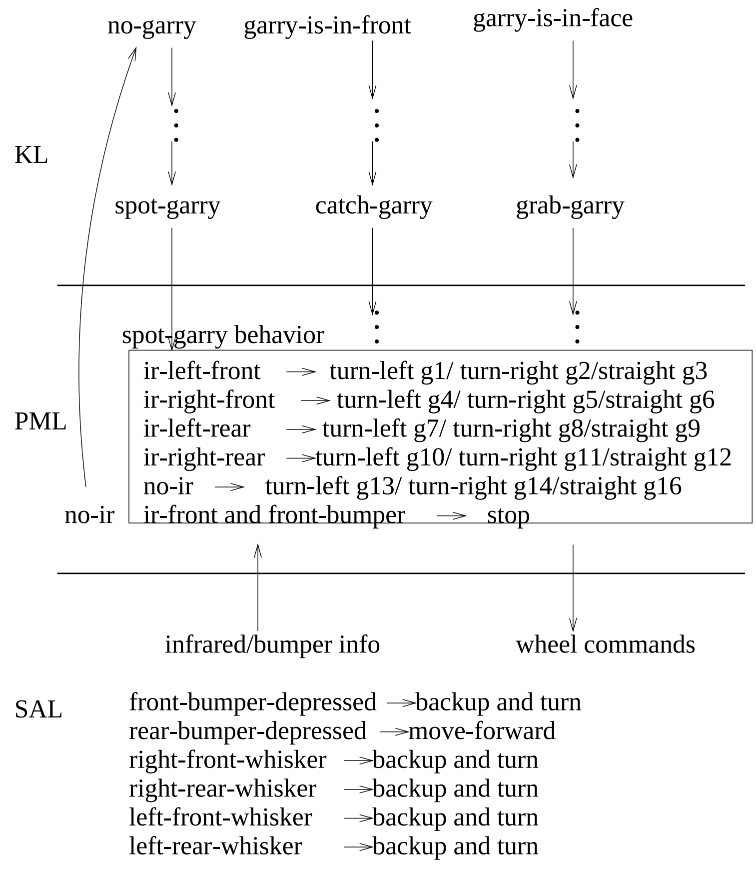

Figure 5.4: Schematic representation of Gerry's GLAIR components. The box in the PML shows a PMA for spotting Gerry; there are others for catching up with Garry and grabbing Garry. gi in the PMA box next to each action represents goodness of action in the context of corresponding situation.

At the SAL, Gerry has 6 reflexes for avoiding obstacles. I have programmed PMAs and the knowledge at the KL so Gerry will look for Garry and will stop when it finds Garry. At the PML, I am running reinforcement based learning to resolve nondeterminacy. Gerry's actions can be controlled either completely by the KL or by the PML.

5.2.2 Experiment 2: Garry finds the Beacon

Infrared sensors on Garry only sense the presence of an infrared (IR) transmitting source, henceforth referred to as the beacon. In this experiment, Garry is given the task of moving toward the stationary beacon and once there stopping.

IR sensors are placed on the robot and thresholded such that Garry can sense the beacon from the right only, the left only, or the front. When a whisker or a bumper senses contact with an external object, Garry performs a reflex. Reflexes are generated at the SAL and have a high priority. At the PML, Garry has four commands: go-straight, turn-right, turn-left, stop.

Figure 5.5 shows the table top where Garry starts at a random location and orientation. Garry has a single goal of getting to the beacon. Two versions of the table are shown. In the simpler table, there are only two regions. In the more complex table, there are lines as well as regions. Garry can find and orient itself on a line and follow a line to one end.

Garry was endowed with some ability to make sense of its routine interactions and learn from routines. This ability was in understanding its goal in a way that rewards and punishments can be generated and used to improve its skills and learn explicit knowledge of its actions.

The situations used in the PMA are a pair of region color and beacon sensing. The region colors are white and black. The beacon values are *noB* for no beacon, *r* for sensing the beacon from the right only, *l* for sensing the beacon from the left only, and *f* for sensing the beacon straight ahead. For example, *wr* is the situation where Garry is on a white region and sees the beacon on its right side only. When Garry is on the farther black region on the more complex table, the situation sensed is *bnoB*. This is because the beacon range is only to the middle of the table. *bnoB* indicates that Garry is on a black region and the beacon is not visible. A distinguished situation is being at the beacon denoted by *a*. This is when Garry is facing the beacon, standing on the half-circle white area, and has its front bumper depressed. Garry's command transitions have situation and behavior pairs as antecedents and four commands as consequents.

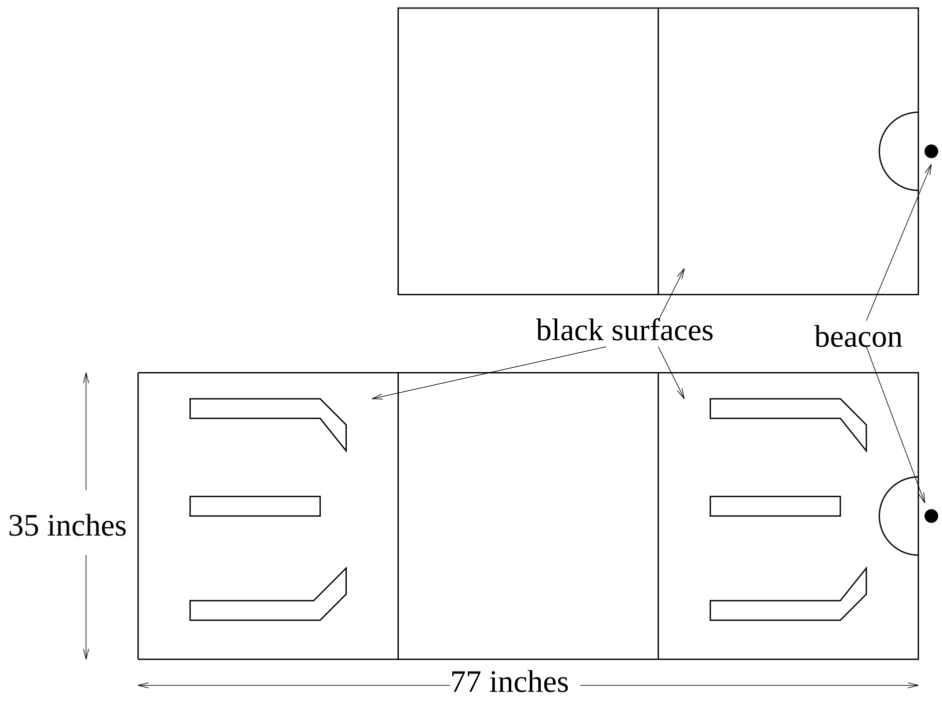

Figure 5.5: Table top for Garry finding the beacon. All regions on the table are white unless indicated to be a black surface.

Figure 5.6 shows Garry's goal sketch *to go to beacon* at the KL. *to go to beacon* is decomposed to two desirable subgoals: (1) *orient toward beacon*, and (2) *move toward beacon*. The goal sketch specifies that as long as the beacon is visible, any region transition is desirable since it gets Garry closer to the beacon.

The leaf nodes of the goal sketch are the following six nodes. r is Garry's goal to see the beacon by its right IR sensor when it can't see the beacon at all. l is Garry's goal to see the beacon by its left IR sensor when it can't see the beacon at all. f is Garry's goal to see the beacon by both its IR sensors when it either can't see the beacon at all or sees the beacon only from one side. *b(lrf)* denotes Garry's goal to repeat a situation in successive PML cycles. The situation to be repeated in *b(lrf)* is: being on a black region, seeing the beacon either from its left side, from its right side, or from straight ahead. *(f)* denotes Garry's goal to repeat the situation of seeing the beacon by both its IR sensors in successive PML cycles. a is Garry's goal to be at the beacon.

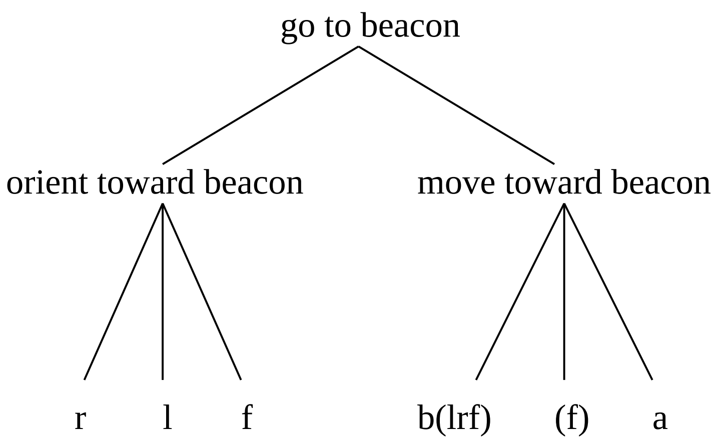

Figure 5.6: Garry's Goal Sketch to go to beacon consists of two goals of moving toward the beacon, and orienting toward the beacon. There are six leaf goal nodes.

Chapter 6

Partial Implementation of the Theory

6.1 Implementation of the Process of Automaticity

The aim of this section is to illustrate the process of automaticity using my experiments. Perceptual, mental, and physical tasks can be subjects of automaticity. In this section, I consider knowledge of "what to do next" [Hexmoor et al., 1992] (i.e., given the current situation, what action should be performed) in order to illustrate automaticity. Automaticity can be applied to cognitive processes other than the knowledge of "what to do next", but in this section I will not consider cognitive processes in general. In GLAIR, migration of knowledge about actions from the KL to the PML is a form of *automaticity*. I use Gabby to illustrate automaticity. In the remainder of this section, I describe an implementation of knowledge migration in Gabby. Migration transforms the representation of knowledge from the KL to the PMA format.

When Gabby determines an action in the KL, it submits a pair of a primitive action A and a complex act G to the PML. In Gabby, there is a goal associated with each complex act. At the PML, commands are associated one-to-one with KL actions, and behaviors are associated one-to-one with KL complex acts. Consider C and B to be the unique counterparts of A and G, respectively, and S to be the current situation. If $S \times B \mapsto C$ is not a CT, it is made one. B is the behavior in effect, and unless a new complex act is selected at the conscious level, B remains the behavior in effect. If a new G (corresponding to new behavior B' in the PM level) is transmitted to the PM-level, the behavior in effect is updated to be B'. If $S \times B \mapsto B'$ is not a BT, it is made one.

I started Gabby with an empty PMA, and as the game was played, PMA transitions were learned. While PMA transitions were being learned, Gabby used PMA transitions if appropriate PMA CTs were already learned. Otherwise, the KL handled command generation. As the game was played, I observed that the agent became more reactive, since the PMA was used to generate behaviors more often than the knowledge level. Figure 6.1 shows that over the course of successive games, Gabby used PML more and used KL less. The data for this Figure were collected over 500 games.

Figure 6.2 shows a small sample of the PMA transitions that Gabby learned while playing ABS. See Appendix 4 for all the transitions Garry learned at the PML. If the behavior is to catch-up, and the situation of the opponent is (close left, close front, close below, and same-heading), then according to bt1 the effective behavior will change to maintain attacking. With maintain attacking as the effective behavior, if the situation persists, according to ct1, the best command will be to keep altitude, turn right, and normal speed. On the other hand, if the situation is (close right, close front, close above, and same heading), according to ct2, the best commands will be one of ascend, maintain heading, normal speed or keep altitude, turn left, normal speed. If the situation is (close left, close front, far behind, and same heading), with behavior of catch-up, the best command in ct3 is descent, maintain heading, and go fast. With catch-up as the effective behavior, the situation (far right, close front, close below, and same heading) is the best command in ct4.

The best commands in ct1 and ct2 might seem counterintuitive but Gabby has learned that when the opponent is being attacked, the opponent will try an evasive maneuver. So unconsciously anticipating the opponent's evasive maneuver, Gabby tries to maintain chasing the opponent by turning.

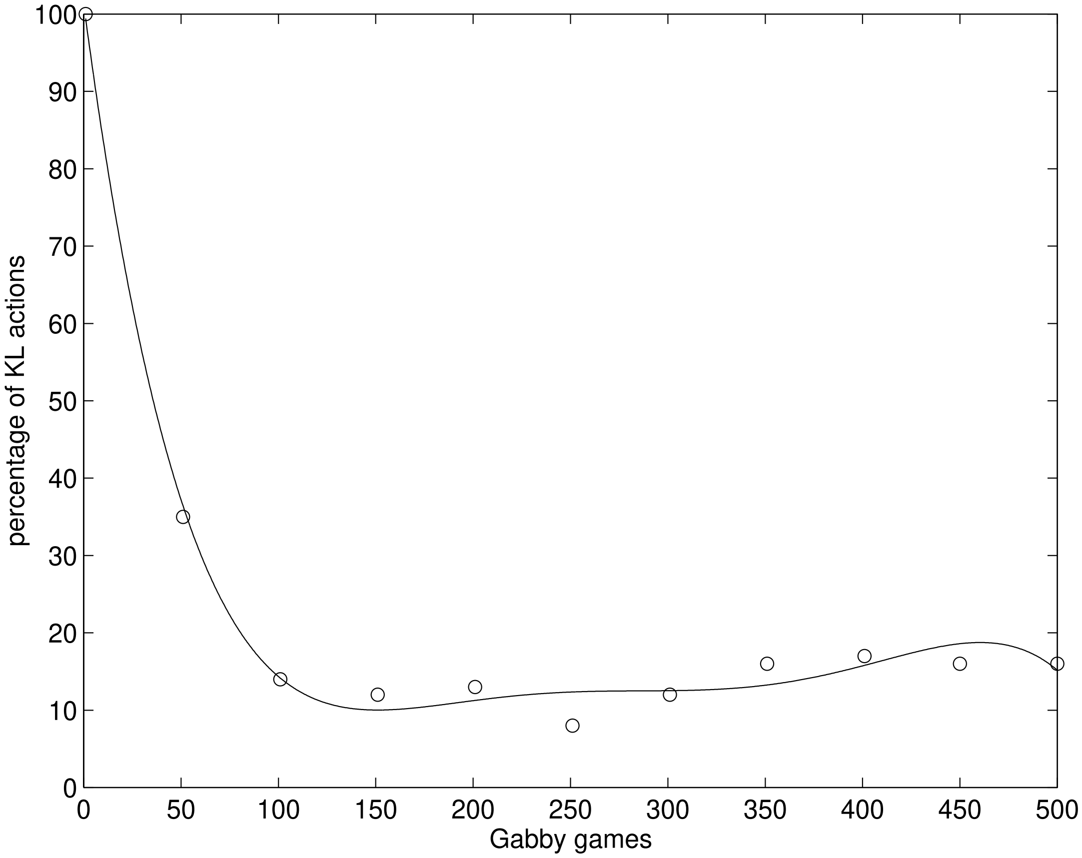

Figure 6.1: Rate of automaticity: In the first game, Gabby actions came entirely from KL. Since it is so close to zero, this Figure does not mark the first game. For the 50th game, 35 percent of the Gabby's actions came from KL. After about 100 games, Gabby settles to having about 16 percent KL actions.

6.2 Implementation of the Process of Habituation and Skill Refinement

The aim of this section is to illustrate the process of habituation and skill refinement using my experiments. Perceptual, mental, and physical tasks can be subjects of habituation and skill refinement. In this section, I consider motor actions for habituation and skill refinement. In GLAIR, this process is entirely at the PM-level. First, I will use Gabby and reinforcement-based learning to illustrate habituation and skill refinement. Next, I will use Garry and goal sketches to illustrate habituation and skill refinement.

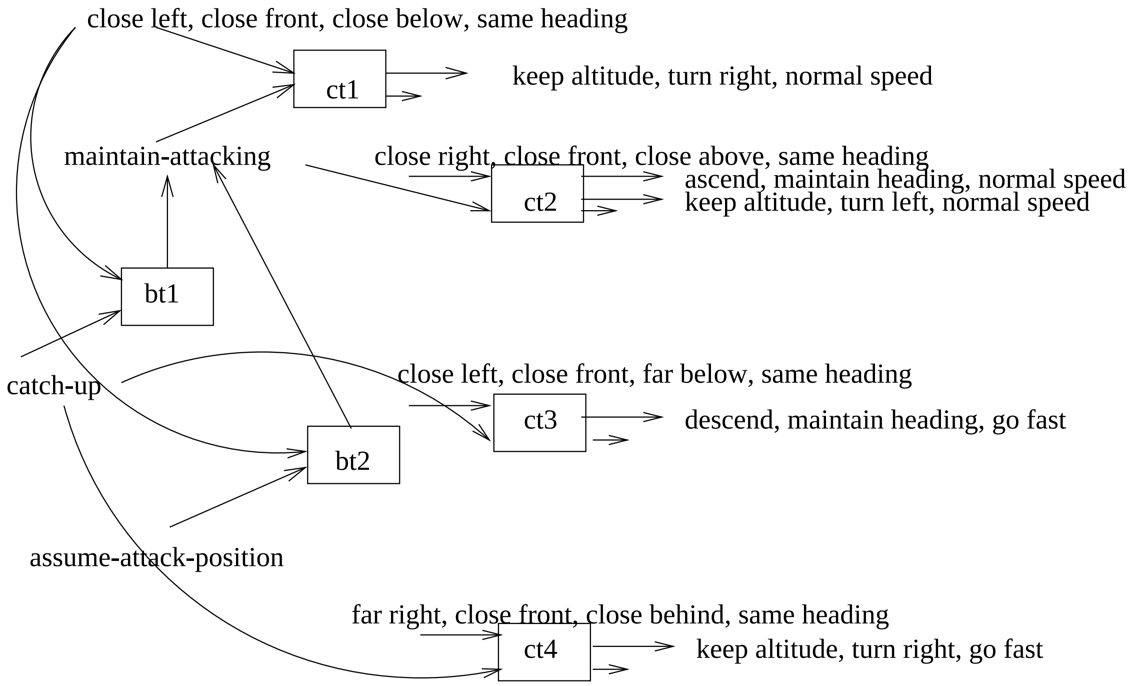

Figure 6.2: Sample of Learned Command and Behavior Transitions

I experimented with my proposed extensions to reinforcement based learning as they are discussed in the chapter 4. I did not find any improvements to Gabby's game playing, i.e., no skill refinement took place. I do not discount my extensions to reinforcement based learning. However, I did not find Gabby or my other agents to be good candidates for demonstrating my extensions. Therefore, I leave this to future work.

6.2.1 Gabby and Reinforcement-Based Learning

The aim of learning in Gabby is to learn which commands, when associated with a situation, yield a better result, i.e., Gabby ends up in a more desirable situation. Some situations in ABS are more desirable for Gabby than others, e.g., being right behind the enemy and in shooting range. I assign a reward value $\mathbf{R}(s)$ between -1 and 1 to each possible situation S. As Gabby makes a move, it finds itself in a new situation. This new situation is not known to Gabby beforehand, since it also depends on the other pilot's move. Since the new situation is not uniquely determined by Gabby's move, Gabby's view of the game is not

Markovian.[1]

I used Q learning [Watkins, 1989], which is a particular variety of reinforcement based learning.[2] In Q learning, **Utility**(S(t),C(t)) is the evaluation of how appropriate command C is in situation S when C is executed at time t in response to S at t. **R**(S(t+1)) is the reward received by being in state S(t+1). For the agent, rewards are determined as the game is played and can not be determined beforehand. This is called the immediate reward. The following equality (shown first in chapter 4 and repeated below for ease of reference) is maintained and propagated as each command is executed. At the start of the game, all **Utility**(S,C) in PMA are set to 1.

$$\mathbf{Utility}(S(t), C(t)) = \mathbf{R}(S(t+1)) + \gamma[\lambda \mathbf{Utility}(S(t+1), C(t+1)) +$$

$$(1-\lambda) max_{C(t+1)} \mathbf{Utility}(S(t+1), C(t+1))]$$

To test reinforcement based learning, I developed a programmed opponent for Gabby that plays according to a fixed strategy, written to simulate the strategy a human player might use. I had the two programs play 25 games without learning. Gabby won about 30% of these games. I then enabled Gabby's reinforcement based learning, and had the programs play an additional 197 games. On the average, each game consisted of about 25 moves for each player. Gabby learned with each move and used what it learned from one move to the next move and from one game to the next game. With the learning mode on, I measured the ratio of winning to losing games for each set of 25 games and observed that Gabby's ratio of winnings improved. After 75 games Gabby reached a ratio of 50% winning. After that, Gabby's winning ratio didn't improve any further. Figure 6.3 shows the improvement in Gabby's game playing ability in 175 games.

In the following section, I use Garry to continue demonstration of habituation and skill refinement. Instead of reinforcement-based learning I used in Gabby, Garry uses a

[1] The Markovian property is that the current state (no memory) is all that is needed to produce an optimal action policy. The Markovian assumption for an agent is that the effect of an agent's actions only depends on its current state.

[2] Q-learning is mathematically proven to be effective in Markovian environments [Watkins, 1989]. I have successfully applied reinforcement-based learning in the non-Markovian domain of ABS. Proof of why reinforcement-based learning was successful for Gabby would make a good analysis but it is beyond the scope of my dissertation.

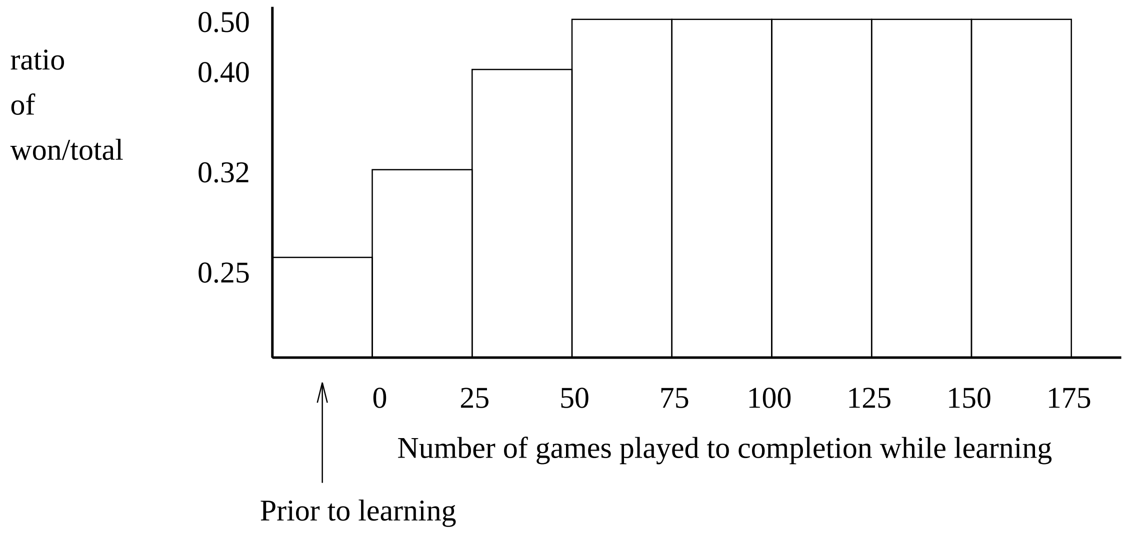

Figure 6.3: Learning rate using reinforcement based Q learning.

goal sketch to demonstrate habituation and skill refinement. There is no conflict between reinforcement-based learning and goal sketch and can be used in conjunction.

6.2.2 Garry and its Goal Sketch

The goal sketch manager uses the goal sketch at the KL to generate rewards and punishments by matching the existing conditions to goal-sketch nodes.

It is obvious from the setup of Garry that it needs to learn biases in command transitions in order to prefer commands that make it see the beacon and to avoid commands that will lose track of the beacon. Similarly, behavior transitions that make Garry approach the beacon will have to be preferred over the ones that don't. Command goodnesses are a direct indication of the level of learning.

Figures 6.4 to 6.11 show the effect of Garry's learning after about 2000 PMA cycles, which adds up to 4 hours of real-time experimentation. The figures show all command transitions and how, after each reward, probabilities among commands in a command transition are changed. In each figure, CT is a command transition, situation and behavior are shown to the left of the CT box and commands with their corresponding probability of selection are shown to the right of the CT box. In some CTs, there is only right choice for a command. For example, in Figure 6.5 Gabby correctly chose to turn left. In contrast to a single correct move, in some cases there are multiple good choices. For example, in Figure 6.4 Garry could have learned either to turn right or to turn left to see the beacon. However, Gabby chose turning right and stayed with that choice. This demonstrates habituation.

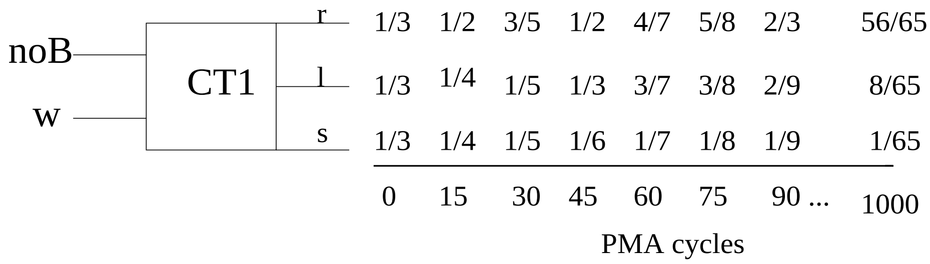

Figure 6.4: Probability of choosing a command for the CT where Garry is on a white region and doesn't see the beacon. In this and following 7 figures, I have plotted probabilities of choosing command r, l, and s along PMA cycles. The 1000 PMA cycles spans many runs of finding the beacon from start to finish.

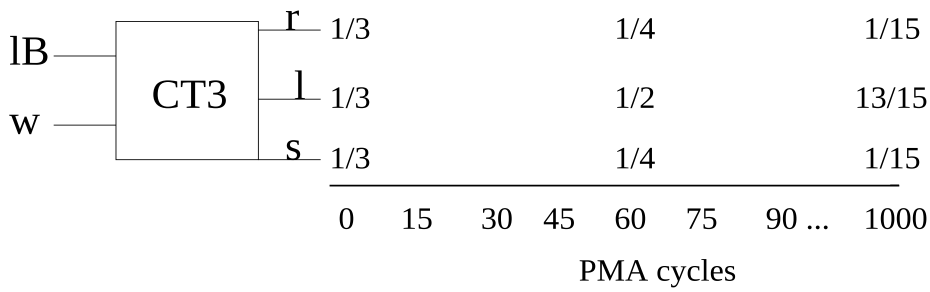

Figure 6.5: Probability of choosing a command for the CT where Garry is on a white region and sees the beacon on its left

Although I want GLAIR-agents to learn from consequences of their actions in the PML in improving their performance, I don't want them to learn the any long-term consequences explicitly in the PML. For example, Gabby did not know about long-term consequences of its actions. Even though it improved in the game, it did not represent the consequences in the PML. I also don't want GLAIR-agents to completely rely on unconscious learning. In order to learn more about actions, it is necessary for the agents either to explicitly reflect on their interactions with the world or to have an automatic self-observation mechanism. I am particularly interested in having GLAIR-agents learn sequences of actions that are in no way guaranteed to be successful but that are commonplace for the agent and have been

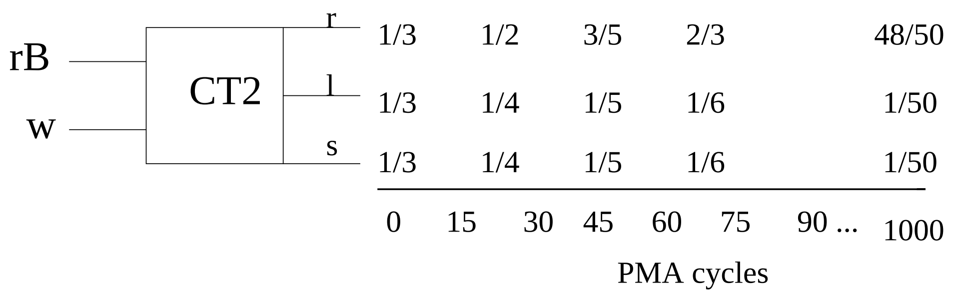

Figure 6.6: Probability of choosing a command for the CT where Garry is on a white region and sees the beacon to its right

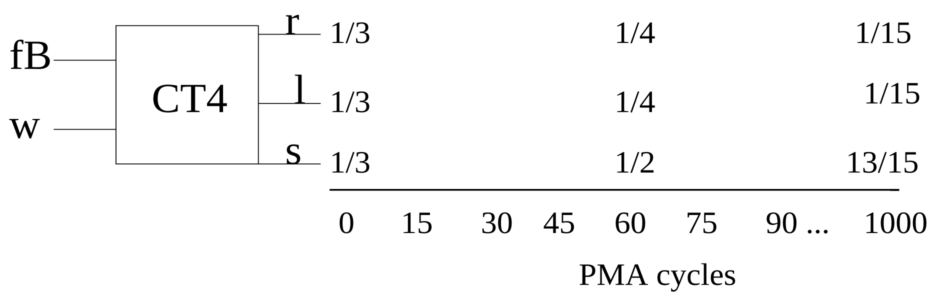

Figure 6.7: Probability of choosing a command for the CT where Garry is on a white region and sees the beacon in front (i.e., both left and right)

used frequently with a high degree of success. In the following section, I discuss learning explicit action preconditions, action consequences, and action sequences.

6.3 Implementation of the Process of Discovering New Knowledge Chunks

The aim of this section is to illustrate discovery of new relationships among knowledge pieces. Perceptual, mental, and physical tasks can be subjects of discovery. In this section, I consider learning preconditions, effects, and knowledge about disconnected pieces of knowledge. In GLAIR, this process is at the KL, with influences from the PML. Gabby and Garry illustrate discovery.

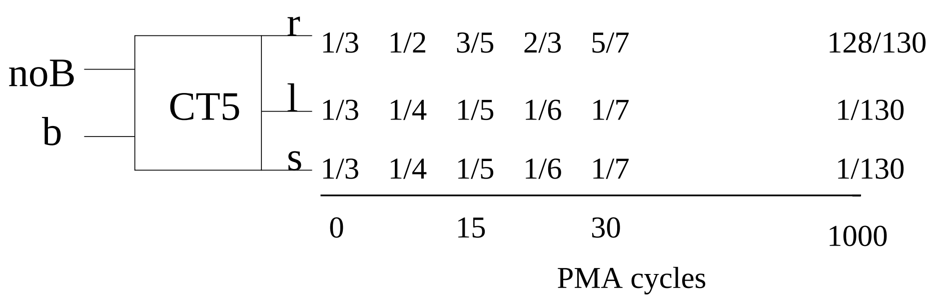

Figure 6.8: Probability of choosing a command for the CT where Garry is on a black region and doesn't see the beacon

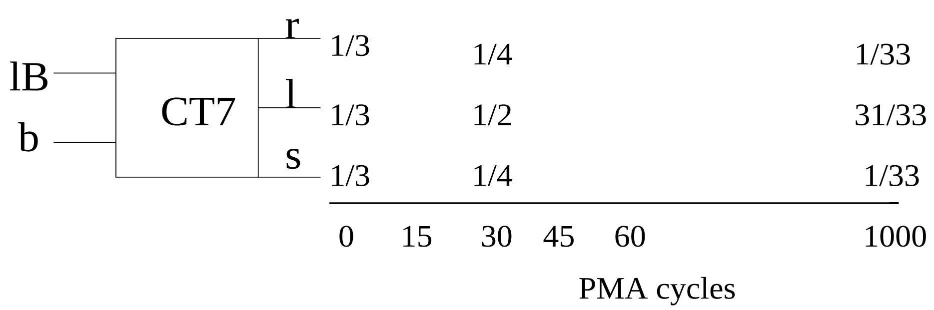

Figure 6.9: Probability of choosing a command for the CT where Garry is on a black region and see the beacon to its left

6.3.1 Gabby

Figure 6.12 shows the frequency of successive behavior selection by Gabby in about 500 games played to completion. This data is collected from section A.2. The table is constructed as an adjacency data structure, i.e., given a row, the numbers in that row represent how many times the behavior at the left of the row was followed by the behavior in top of the column. For example, shoot-chasing-opponent was followed by attack 78 times.

I appeal to intuition that the data in my table reflects an aggressive WWI "dog-fight" strategy. Commonsensically, I believe a WWI "dog-fight" pilot would be "conscious" of his aggressive "dog-fight" strategy. The pilot could "consciously" describe the "unconscious" behavior transitions he uses in flight, i.e., the pilot somehow retrieves "unconscious" data.

Gabby started out with pieces of knowledge at the KL corresponding to behaviors in Figure 6.12.[3] Behaviors and transitions were learned at the PML. Whereas successive

[3] See Appendix A for meaning of behaviors

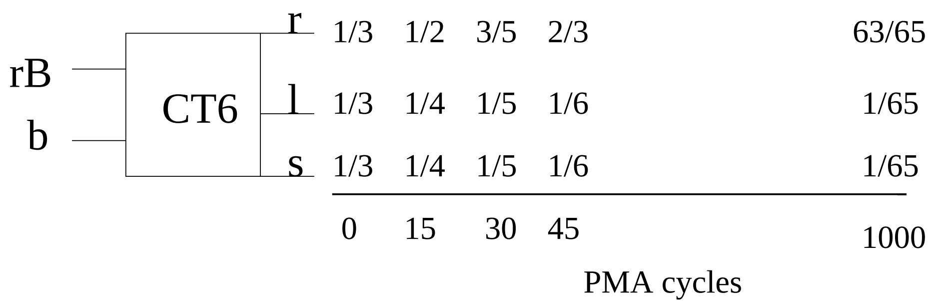

Figure 6.10: Probability of choosing a command for the CT where Garry is on a black region and sees the beacon to its right

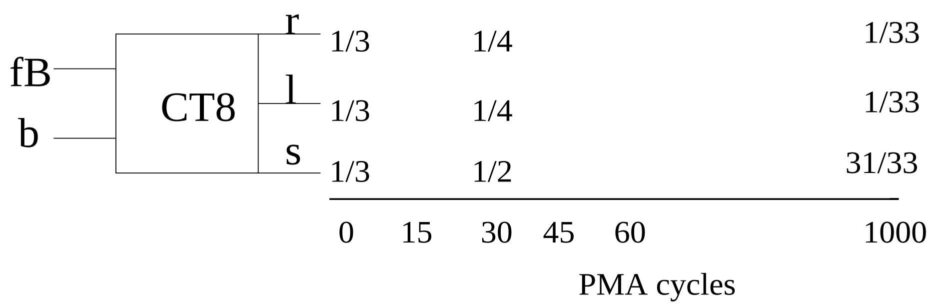

Figure 6.11: Probability of choosing a command for the CT where Garry is on a black region and sees the beacon in front (i.e., both left and right)

behavior frequencies are available at the PML, Gabby does not know the frequency of the corresponding successive knowledge that is used at the KL. I argue that, similar to the WWI pilot, we can use the frequency of behavior transitions at the PML to construct knowledge about previously disconnected pieces of knowledge that were used to construct the behaviors.

I am suggesting that (a) Gabby be allowed to retrieve certain "unconscious" behavior successions, (b) Gabby be able to explicitly represent that succession, and (c) Gabby be able to communicate that knowledge. "Attack" is the most commonly selected behavior and six out of eight behaviors are followed by "attack." Therefore, the fact that "avoid-attack" is followed by "attack" is less interesting than, for example, the fact that "attack" is being followed by "shoot-chasing-opponent." I want Gabby be able to tell us that "attack" is followed by "shoot-chasing-opponent" and "avoid-crash" is followed by "avoid-attack."

Prior to the learning that I described in this section, Gabby's knowledge of behaviors

	catch-up	attack	avoid-attack	avoid-opponent	avoid-crash	shoot-chasing-opponent	maintain-attacking	assume-attack-position
catch-up	0	27	3	0	0	0	1	0
attack	4	0	4	8	1	17	1	2
avoid-attack	1	41	0	0	0	0	0	2
avoid-opponent	4	23	0	0	0	8	0	0
avoid-crash	0	1	4	3	0	0	0	0
shoot-chasing-opponent	0	78	4	0	0	0	0	0
maintain-attacking	7	32	0	0	0	0	0	2
assume-attack-position	2	19	4	1	0	0	1	0

Figure 6.12: This table shows the frequency of successive behavior selection by Gabby in about 500 games

is disparate and there is no relationship between behaviors. In this section, I made a point about what can be learned at the KL by Gabby. In my experiment, I did not add the knowledge about behavior succession and I did not demonstrate how Gabby can use that knowledge. I leave further experimentation with this knowledge to future work.

6.3.2 Gerry and Garry

In the experiment where Gerry chases Garry, Gerry repeats three behaviors of *Spotting Garry*, *Catching up with Garry* and *Grabbing Garry*. Similar to Gabby's learning of sequences, I suggest that Gerry be allowed to learn that these three behaviors happen in succession. As in Gabby, I did not add the knowledge about behavior succession to the KL and I did not demonstrate how Gerry can use that knowledge. I leave further experimentation to future work.

Garry was developed after my experiments with Gerry and Gabby. At this point, I had developed ideas about the goal sketch as a model for knowledge discovery. Using the goal sketch at the KL, the following very simple effect propositions are learned at Garry's KL: *rlf is an effect of "wander on white"*, *rlfa is an effect of "wander on black"*. The first proposition states that an effect of Garry's wandering on white is seeing the beacon from its right, left, or front. The second proposition states that an effect of Garry's wandering on black is seeing the beacon from its right, left, or front, or being at the beacon. (The abbreviations for situation symbols were given in chapter 5.)

The following precedence propositions are learned at the KL: *wrlf precedes "wander on black"*, *brlf precedes "stopping at the beacon"*. The first proposition states that Garry will be on a white region and see the beacon from its right, left, or front before wandering on black. The second proposition states that Garry will be on a black region and see the beacon from its right, left, or front before stopping at the beacon. Finally, Garry learns that *"wander on white" precedes "wander on black"* is a sequence of actions it performs repeatedly.

Chapter 7

Conclusion

This dissertation shows that routine activities in the world of an autonomous agent are a major source of learning for the agent. In my approach, an agent interacts in the world in several different ways, from cognitive to automatic. I showed that an agent can learn and also improve its routine interactions in its different modes of interaction in the world.

I proposed a theory that sketches four processes affecting the "mind" of an agent engaged in routine interactions: (1) automaticity, (2) habituation and skill refinement, (3) abstraction by chunking, and (4) discovery of new knowledge chunks.

The significance of my theory is that it attempts to model cognitive and sub-cognitive processes involved in an intelligent agent engaged in routine activities. My model helps enrich the agent's internal representations and abilities. My theory synthesizes theories in human psychology and methods in machine learning. I have developed a few agents and experiments that partly exhibit my theory.

Gabby is the agent in a WWI air battle experiment called Air Battle Simulation. Gabby exhibits automaticity by caching into a flat stimulus-response data structure its explicit knowledge about actions to catch up with and shoot the enemy as well as escape being chased. Gabby also exhibits habituation and skill refinement. For improving its maneuvering, Gabby uses rewards and punishments for being in good and bad relative positions with respect to the enemy. Using rewards and punishments, Gabby improves its maneuvers.

Gabby starts out with a set of disconnected pieces of explicit knowledge controlling its interactions. As Gabby caches its knowledge into a flat stimulus-response data structure, the transitions between disconnected knowledge pieces are recorded in the flat data structure. My theory argues for making explicit (i.e., discovering) succession of behaviors as action sequences. This in part demonstrates my arguments about discovering new knowledge chunks.

Gerry and Garry are two robot agents. In one experiment, Garry learns to locate and move to a beacon on a table top. Garry exhibits habituation and skill refinement by using environmental clues as rewards. I used a concept first introduced in psychology, the goal sketch, as a representational mechanism to encode rewards from environmental clues. In another experiment, Gerry chases Garry and learns to build new relations with its knowledge. This is a demonstration of discovering new knowledge chunks.

Gabby, Gerry and several other agents I developed are based on an agent architecture I developed and called GLAIR. GLAIR agents are capable of reasoning, deliberative actions, natural language interaction, learning, and reactive interaction. Each of GLAIR's three levels (Sensori-actuator, Perceptuo-motor, and knowledge) uses different representations and models different components of intelligent agency. GLAIR levels operate semi-autonomously. Whereas intra-level learning improves an agent's performance, inter-level learning migrates know-how from one level to another. I briefly discuss a few of the lessons learned from my agents as conclusions in the remainder of this chapter. For more details see [Hexmoor, 1995b].

In GLAIR's real-time environments, concurrent and decentralized mechanisms with overlapping problem spaces proved useful for generating behaviors. Having a fast reflexive layer as well as slower, more deliberative layers running concurrently is similar to the human attention mechanism and dealing with situations at different levels of abstraction.

Experiments with Gabby demand fast execution of skills. This shows us that skills must be represented in data structures that makes their access fast. This is in contrast to knowledge which requires a data structure that maintains a chain of reasoning. This confirmed the idea that knowledge and skills should be represented with different formalisms.

It is obvious that we need agents to learn autonomously and explicitly by reasoning and acquiring knowledge, and implicitly by improving their interactions. I explored goal sketch as a mechanism to specifies clues and conditions for task completion independent of actions. I developed a mechanism for a goal sketch using agent to identify relevance of events in the world to the agent's actions and goals. I demonstrated the usefulness of goal sketch in my experiments with Garry by enriching Garry's knowledge about actions. I conclude that we need to continue the development of mechanisms to understand goals and to identify relationships between events in the world and agent actions.

Chapter 8

Future Work

8.1 Continuing Demonstrations of Theory for Routines

I intend to demonstrate abstraction and chunking with my experiments. Perceptual, mental, and physical tasks can be subjects of abstraction and chunking. In GLAIR, the process of abstraction and chunking demonstrates the effect of increased use of PML on the KL.

I have theorized that routine interactions are a source of discovering new knowledge. I used a psychological concept of the goal sketch in my theory. However, the concept of goal sketch is very sketchy, and I have only begun to formalize it in explaining everyday discoveries. Significant work remains in further defining the goal sketch, how it interacts with agent's knowledge of actions and plans, and how to make use of it in knowledge discovery. I assigned a reward with each leaf node of the goal sketch and used those rewards for habituation and skill refinement at the PML. I suggest using the very same rewards as a goal sketch metric. A given pattern of commands that satisfies a goal sketch will satisfy the leaf nodes of a goal sketch. I propose adding all the rewards from satisfaction of goals and using that value for deciding whether a given pattern is a good strategy.

In my theory I proposed an extension to reinforcement based learning that takes into account an element of habituation. However, this is not demonstrated in my experiments. I will demonstrate this in my future work.

I have not yet explored possible relationships between the four processes in my theory.

For example, I believe there is a strong relationship between chunking and automaticity. I believe these might be part of a larger process and in that larger process, chunking affects KL and automaticity affects PML. I leave this to future work.

8.2 Future work in GLAIR

PMA is a system that I developed to model and automatically learn reactive motor skill interactions of an agent with the world. As such I have used PMA in a handful of experiments. However, I haven't argued adequacy of PMA in modeling all motor skills. Also, I haven't fully explored other robot learning modes that can be implemented in PMA. PMA is very similar to perceptron like systems in [Nehmzow, 1992]. I plan rigorously to compare the two systems and apply them to the same example.

GLAIR is an architecture for modeling intelligent agency. So far, I have used GLAIR in building synthetic agents that perform their task intelligently and improve their interaction through learning. I would like to extend my implementations to systems where I can have the synthetic agents interact realistically with live subjects. This will help in psychological validation of GLAIR.

8.3 More GLAIR Agents

FEVAHR is a GLAIR-based mobile robot agent I have implemented in collaboration with professor Stuart C. Shapiro and Amherst Systems Inc.[1] FEVAHR uses foveation to guide its visual servoing on target and obstacles while the robot locates and follows another robot or other visually distinct objects. I have been exploring habituation and skill refinement for visual gaze control and navigation in FEVAHR. I will explore other parts of my theory about routines in FEVAHR.

ABS can be extended to include a wider range of basic maneuvers like *banking* which will make the game more realistic. This will give us more opportunities to test learning ideas in the flight domain. Another improvement to ABS will be to include a realistic time

[1]FEVAHR stands for Foveal Extra-Vehicular Activity Helper-Retriever. EVAHR is the title of the NASA Johnson Space Center's Automation and Simulation Division project whose objectives subsume those of FEVAHR.

element. Playing against a human being with realistic times will give us an opportunity to test human reactions and to further validate the GLAIR model.

As small agents, Gerry and Garry, are limited in their sensing and computational abilities. A version of computer control boards used on these robots can be used to build a flying agent that learns motor skills of flying in routine flight maneuvers.

A GLAIR-based agent that I partly developed but did not use for demonstrating my theory about routines is called the Robot Waiter, RW. RW is a robot arm designed to set a dinner table in various configurations, guided by input from a color camera and controlled by a host computer. Among other things, RW was designed to discover that stacking dishes before carrying them is easier than making many trip to the cupboard. For reasons beyond my control the robot arm developed mechanical problems and I postponed the project. Issues related to routines involved in the task of setting a table are interesting and I plan on pursuing these in the future. The interested reader can see more details in [Hexmoor et al., 1993b].

Another GLAIR-based experiment that I partly developed but did not use for demonstrating my theory about routines is called the Mobile Robot Lab. The agent in this experiment was not given a name. The room in which the robot moves has a polygonal floor plan and vertical walls, and contains a number of solid objects with convex polygonal bases and vertical faces, each with an associated user-defined spectral reflectance function (with a 10 nm resolution), which together with a lighting and camera model determines the perceived color of objects. Among other concerns, this agent was developed to explore learning new concepts about mobility and pushing objects. With the development of Garry and Gerry, this project was postponed. Issues related to routines involved in the task of mobility and box pushing are interesting and I will use this environment in future work on routines. The interested reader can see more details in [Hexmoor et al., 1993b, Hexmoor et al., 1993c].

Computer video games that require coordination skills and strategy development are well suited to demonstrate my theory about routines. For example, the simple game of Minesweep commonly available on PCs is a good task for a GLAIR agent. This is a game that requires both reasoning and skill and it is played with a countdown timer. I plan to explore automaticity, chunking, and knowledge discovery using a video game.

Appendix A

Transitions Gabby learned at the PML

A.1 Command Transitions

The following is a list of command transitions. Each transition is a triple of Situation, Behavior, and a command set. For example:

$$((FARLEFT)(FARFRONT)(CLOSEABOVE) <)$$

denotes the situation that the enemy is far to the left, far in front, close above, and heading towards Gabby from the right. Relative orientations are abbreviated as symbols: ^ for flying parallel and heading in the same direction, v for flying parallel but in the opposite direction from Gabby, < for enemy flight direction forming a 90 degree angle to our current heading and heading right to left, and > for enemy flight direction forming a 90 degree angle to our current heading and heading left to right. An example of a behavior is ATTACK.

An example of a command set is

$$((A - R - N0.400000130)(A - SR - N - 0.15000010)(A - SL - N - 0.15000010)$$

$$(K-R-N0.19999990)(K-SR-N0.90000010)(K-L-N1.09999990)$$

$$(K-SL-N0.65000010)(A-M-N1.09999990)).$$

Each command, like A-SR-N, is associated with a goodness level (-0.1500001) indicating the likelihood of Gabby choosing that command. The following abbreviations are used in the commands: A for Ascend, K for keep, D for descend, L for left, R for right, SR for shift right, SL for shift left, M for maintain (heading), F for fast, N for normal speed, and C for cut (engine).

In order to ease understanding of transitions, I give Gabby's common-sense meaning for the behaviors. I descibe when each behavior is used and what Gabby does with each behavior.

- avoid-attack: when being attacked by the opponent in close-range; turn left or right, up or down and go fast.
- avoid-opponent: when flying head on to the opponent but still far away; considering the relative orientations with the opponent turn so as to end up behind the opponent.
- avoid-crash: when flying head on to the opponent in close-range; turn left or right fast.
- maintain-attacking: when flying behind the opponent (close or far); move forward (this will be any movement and turn that will maintain distance or improve the distance to the opponent).
- attack: when the opponent is behind Gabby (close or far); considering the relative orientations with the opponent turn so as to end up behind the opponent.
- catch-up: when the opponent is far ahead; move forward fast (this may include turns).
- assume-attack-position: when the opponent is close in front; turn to match the direction of the opponent.
- , shoot-chasing-opponent: when the opponent is far behind but is catching up; turn the engine off for a slow dive (this might include a turn).

This list shows the CTs after 500 games.

```
(
(((FAR LEFT) (FAR FRONT) (CLOSE ABOVE) <) ATTACK ((K-L-F -0.6999999 0)))
(((FAR RIGHT) (FAR FRONT) (CLOSE BELOW) <) ATTACK ((K-R-F -0.2 0)))
(((CLOSE LEFT) (FAR BEHIND) (FAR BELOW) >) ATTACK ((K-L-F 0 0)))
(((CLOSE LEFT) (FAR FRONT) (FAR ABOVE) >) ATTACK ((K-SL-F 0.2 0)))
(((FAR LEFT) (CLOSE FRONT) (FAR ABOVE) ^) CATCH-UP ((A-M-F -0.2 0)))
(((FAR RIGHT) (CLOSE FRONT) (CLOSE BELOW) <) ATTACK ((K-R-F 0.45000008 0)))
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR BELOW) ^) SHOOT-CHASING-OPPONENT
 ((K-R-C -1.35 0)))
(((CLOSE RIGHT) (CLOSE FRONT) (FAR BELOW) ^) CATCH-UP ((D-M-F -1.3 0)))
(((CLOSE RIGHT) (CLOSE FRONT) (CLOSE ABOVE) ^) MAINTAIN-ATTACKING
 ((A-R-N 0.40000013 0) (A-SR-N -0.1500001 0) (A-SL-N -0.1500001 0)
  (K-R-N 0.1999999 0) (K-SR-N 0.9000001 0) (K-L-N 1.0999999 0)
  (K-SL-N 0.6500001 0) (A-M-N 1.0999999 0)))
(((CLOSE RIGHT) (CLOSE FRONT) (CLOSE ABOVE) >) ASSUME-ATTACK-POSITION
 ((K-R-N -0.8999999 0) (K-SR-N 0.34999996 0)))
(((CLOSE RIGHT) (CLOSE FRONT) (CLOSE ABOVE) V) AVOID-CRASH
 ((K-R-F 0.05000007 0) (K-L-F 0.05000007 0)))
(((CLOSE RIGHT) (CLOSE FRONT) (CLOSE ABOVE) <) ASSUME-ATTACK-POSITION
 ((K-SR-N 0.20000005 0) (K-R-N -1.4499999 0)))
(((CLOSE RIGHT) (CLOSE FRONT) (CLOSE BELOW) ^) MAINTAIN-ATTACKING
 ((K-M-N -0.79999995 0) (K-R-N 0.49999997 0) (D-L-N 1.0999999 0)
  (A-R-N 0.100000024 0) (D-SR-N 1.5999999 0) (D-M-N 0.5999999 0)
  (D-SL-N 0.5999999 0) (K-SR-N 1.8 0)))
(((CLOSE RIGHT) (CLOSE FRONT) (CLOSE BELOW) >) ASSUME-ATTACK-POSITION
 ((K-SR-N -0.45000005 0) (K-R-N -2.35 0)))
(((CLOSE RIGHT) (CLOSE FRONT) (CLOSE BELOW) V) AVOID-CRASH
 ((K-L-F 0.3 0) (K-R-F -0.5999999 0)))
(((CLOSE RIGHT) (CLOSE FRONT) (CLOSE BELOW) <) ASSUME-ATTACK-POSITION
 ((K-R-N -0.5999999 0) (K-SR-N -0.5999998 0)))
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE ABOVE) ^) MAINTAIN-ATTACKING
 ((A-SR-N 0.3 0) (A-R-N 0.65000004 0) (A-M-N -0.5999998 0)
  (K-SR-N -1.0999999 0) (K-SL-N 0.05000007 0) (K-R-N 0.9499999 0)
  (A-SL-N -0.35000002 0) (A-L-N -2.1499999 0)))
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE ABOVE) >) ASSUME-ATTACK-POSITION
 ((K-L-N 0.55 0) (K-SL-N -0.099999905 0)))
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE ABOVE) V) AVOID-CRASH
 ((K-R-F -0.099999905 0) (K-L-F -0.05000007 0)))
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE ABOVE) <) ASSUME-ATTACK-POSITION
 ((K-L-N 0.39999998 0) (K-SL-N 0.79999995 0)))
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE BELOW) ^) MAINTAIN-ATTACKING
 ((D-SR-N -0.5999998 0) (D-M-N 0.35000005 0) (K-SR-N 1.5999998 0)
  (K-R-N -0.099999964 0) (D-SL-N -0.20000005 0) (K-M-N 0.34999996 0)
  (D-L-N 0.1500001 0) (K-SL-N -1.25 0) (A-R-N 0.7 0)))
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE BELOW) >) ASSUME-ATTACK-POSITION
 ((K-SL-N 0.099999994 0) (K-L-N 0.7999999 0)))
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE BELOW) V) AVOID-CRASH
 ((K-R-F -0.34999996 0) (K-L-F -0.34999996 0)))
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE BELOW) <) ASSUME-ATTACK-POSITION
 ((K-SL-N 0.09999997 0) (K-L-N 1.3 0)))
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) ^) AVOID-ATTACK
 ((K-L-C -0.4999999 0) (K-R-C -1.6999999 0) (A-R-F -2.4999998 0)
  (A-L-F 0.40000007 0)))
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) >) ATTACK
 ((K-SR-F -0.79999995 0) (K-R-F -0.40000007 0)))
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) V) ATTACK
 ((K-SR-F -0.05000007 0) (K-R-F -0.24999994 0)))
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) <) ATTACK
 ((K-R-F 0.05000007 0) (K-SR-F -0.8999999 0)))
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) ^) AVOID-ATTACK
 ((K-L-C -1.4499999 0) (K-R-C -1.15 0) (A-L-F 0.75 0) (A-R-F 0.29999998 0)))
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) >) ATTACK
 ((K-R-F -0.75 0) (K-SR-F -0.6 0)))
```

```
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) V) ATTACK
 ((K-SR-F -0.3 0) (K-R-F -0.6 0)))
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) <) ATTACK
 ((K-R-F -0.65000004 0) (K-SR-F 1.5499997 0)))
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) ^) AVOID-ATTACK
 ((K-L-C -0.79999995 0) (A-L-F -1.3500001 0) (A-R-F -1.1999999 0)
  (K-R-C -1.5 0)))
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) >) ATTACK
 ((K-SL-F -0.05000007 0) (K-L-F -1.0 0)))
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) V) ATTACK
 ((K-SL-F -0.04999996 0) (K-L-F -0.85 0)))
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) <) ATTACK
 ((K-L-F 0.25000006 0) (K-SL-F -1.0999999 0)))
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) ^) AVOID-ATTACK
 ((K-L-C -0.34999996 0) (K-R-C -0.09999997 0) (A-L-F -0.50000024 0)
  (A-R-F 0.049999982 0)))
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) >) ATTACK
 ((K-SL-F -0.34999996 0) (K-L-F 1.3 0)))
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) V) ATTACK
 ((K-SL-F 0.34999996 0) (K-L-F -0.3 0)))
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) <) ATTACK
 ((K-L-F 0.40000007 0) (K-SL-F -0.45000008 0)))
(((CLOSE RIGHT) (CLOSE FRONT) (FAR ABOVE) ^) CATCH-UP ((A-M-F -0.59999996 0)))
(((CLOSE RIGHT) (CLOSE FRONT) (FAR ABOVE) >) ATTACK
 ((K-SR-F -0.65 0) (K-R-F -1.2 0)))
(((CLOSE RIGHT) (CLOSE FRONT) (FAR ABOVE) <) ATTACK
 ((K-R-F 0.2000001 0) (K-SR-F -1.15 0)))
(((CLOSE RIGHT) (CLOSE FRONT) (FAR BELOW) >) ATTACK ((K-SR-F 0 0) (K-R-F 0 0)))
(((CLOSE RIGHT) (CLOSE FRONT) (FAR BELOW) <) ATTACK
 ((K-SR-F 0 0) (K-R-F 1.75 0)))
(((CLOSE LEFT) (CLOSE FRONT) (FAR ABOVE) ^) CATCH-UP ((A-M-F -1.0999999 0)))
(((CLOSE LEFT) (CLOSE FRONT) (FAR ABOVE) >) ATTACK ((K-L-F 0.45000005 0)))
(((CLOSE LEFT) (CLOSE FRONT) (FAR ABOVE) <) ATTACK
 ((K-L-F 1.4499999 0) (K-SL-F -0.39999986 0)))
(((CLOSE LEFT) (CLOSE FRONT) (FAR BELOW) ^) CATCH-UP ((D-M-F 0.3 0)))
(((CLOSE LEFT) (CLOSE FRONT) (FAR BELOW) >) ATTACK ((K-SL-F -0.05000004 0)))
(((CLOSE LEFT) (CLOSE FRONT) (FAR BELOW) <) ATTACK
 ((K-SL-F -0.20000005 0) (K-L-F 0 0)))
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR ABOVE) ^) SHOOT-CHASING-OPPONENT
 ((K-M-C -0.100000024 0) (K-L-C -0.54999995 0)))
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR ABOVE) >) ATTACK
 ((K-R-F 0.2500001 0) (K-SR-F 0.34999996 0)))
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR ABOVE) V) ATTACK
 ((K-SR-F -0.39999998 0) (K-R-F -1.4999999 0)))
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR ABOVE) <) ATTACK
 ((K-SR-F -0.79999995 0) (K-R-F 0.65 0)))
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR BELOW) >) ATTACK
 ((K-SR-F -0.049999982 0) (K-R-F -1.15 0)))
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR BELOW) <) ATTACK
 ((K-SR-F -1.1999998 0) (K-R-F -0.39999998 0)))
(((CLOSE LEFT) (CLOSE BEHIND) (FAR ABOVE) ^) SHOOT-CHASING-OPPONENT
 ((K-R-C -0.49999997 0) (K-L-C -0.9 0) (K-M-C -2.1999998 0)))
(((CLOSE LEFT) (CLOSE BEHIND) (FAR ABOVE) >) ATTACK
 ((K-L-F 0.1500001 0) (K-SL-F 0.3 0)))
(((CLOSE LEFT) (CLOSE BEHIND) (FAR ABOVE) V) ATTACK
 ((K-L-F 0.25000006 0) (K-SL-F -1.3500001 0)))
(((CLOSE LEFT) (CLOSE BEHIND) (FAR ABOVE) <) ATTACK
 ((K-SL-F -0.54999995 0) (K-L-F -0.6 0)))
(((CLOSE LEFT) (CLOSE BEHIND) (FAR BELOW) >) ATTACK
 ((K-SL-F 0.4000002 0) (K-L-F -1.3499999 0)))
(((CLOSE LEFT) (CLOSE BEHIND) (FAR BELOW) V) ATTACK
 ((K-SL-F -0.45000005 0) (K-L-F 0.14999998 0)))
(((CLOSE LEFT) (CLOSE BEHIND) (FAR BELOW) <) ATTACK
 ((K-SL-F 3.7252903e-8 0) (K-L-F 0.7999999 0)))
(((FAR RIGHT) (CLOSE FRONT) (CLOSE ABOVE) ^) CATCH-UP
 ((K-SR-F -1.6999999 0) (K-R-F -0.19999993 0)))
(((FAR RIGHT) (CLOSE FRONT) (CLOSE ABOVE) >) ATTACK
```

```
 ((K-SR-F 0.049999952 0) (K-R-F -0.14999996 0)))
(((FAR RIGHT) (CLOSE FRONT) (CLOSE ABOVE) V) AVOID-OPPONENT
 ((A-SL-F -0.2999999 0) (A-L-F 0.55 0) (K-L-F 0.30000004 0)))
(((FAR RIGHT) (CLOSE FRONT) (CLOSE BELOW) ^) CATCH-UP
 ((K-SR-F -0.050000012 0) (K-R-F 0.3 0)))
(((FAR RIGHT) (CLOSE FRONT) (CLOSE BELOW) >) ATTACK
 ((K-R-F -0.9999999 0) (K-SR-F -0.35000002 0)))
(((FAR RIGHT) (CLOSE FRONT) (CLOSE BELOW) V) AVOID-OPPONENT
 ((D-L-F -1.6500001 0) (D-SL-F 0.25 0)))
(((FAR LEFT) (CLOSE FRONT) (CLOSE ABOVE) ^) CATCH-UP
 ((K-SL-F -0.8499999 0) (K-L-F 0.04999996 0)))
(((FAR LEFT) (CLOSE FRONT) (CLOSE ABOVE) V) AVOID-OPPONENT
 ((A-SR-F -0.7 0) (A-R-F 0.050000012 0) (K-R-F 0.29999995 0)))
(((FAR LEFT) (CLOSE FRONT) (CLOSE ABOVE) <) ATTACK
 ((K-L-F -0.55 0) (K-SL-F -0.19999996 0)))
(((FAR LEFT) (CLOSE FRONT) (CLOSE BELOW) ^) CATCH-UP
 ((K-L-F -1.1999998 0) (K-SL-F -2.25 0)))
(((FAR LEFT) (CLOSE FRONT) (CLOSE BELOW) >) ATTACK
 ((K-L-F 0 0) (K-SL-F -0.24999997 0)))
(((FAR LEFT) (CLOSE FRONT) (CLOSE BELOW) V) AVOID-OPPONENT
 ((D-SR-F 0.44999996 0) (D-R-F -0.2 0) (K-R-F -0.55 0)))
(((FAR LEFT) (CLOSE FRONT) (CLOSE BELOW) <) ATTACK
 ((K-L-F -0.4 0) (K-SL-F -0.39999998 0)))
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) ^) SHOOT-CHASING-OPPONENT
 ((K-L-C -0.6500001 0) (K-R-C -0.45000005 0) (K-M-C -0.39999998 0)))
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) >) ATTACK
 ((K-SR-F -1.1999999 0) (K-R-F -0.75000006 0)))
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) V) ATTACK
 ((K-R-F -0.24999997 0) (K-SR-F -0.14999996 0)))
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) <) ATTACK
 ((K-R-F -0.19999997 0) (K-SR-F 0.0 0)))
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE BELOW) ^) SHOOT-CHASING-OPPONENT
 ((K-R-C 0.050000012 0) (K-L-C 0.34999996 0) (K-M-C -1.5 0)))
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE BELOW) >) ATTACK
 ((K-SR-F -0.50000006 0) (K-R-F -1.3000001 0)))
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE BELOW) V) ATTACK
 ((K-R-F -0.39999998 0) (K-SR-F 0.20000005 0)))
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE BELOW) <) ATTACK
 ((K-R-F -0.45000002 0) (K-SR-F -0.3 0)))
(((FAR LEFT) (CLOSE BEHIND) (CLOSE ABOVE) ^) SHOOT-CHASING-OPPONENT
 ((K-L-C 0.19999997 0) (K-M-C -1.4901161e-8 0) (K-R-C -0.65 0)))
(((FAR LEFT) (CLOSE BEHIND) (CLOSE ABOVE) >) ATTACK
 ((K-L-F -2.2351742e-8 0) (K-SL-F 0.05000007 0)))
(((FAR LEFT) (CLOSE BEHIND) (CLOSE ABOVE) V) ATTACK
 ((K-SL-F 0.15000007 0) (K-L-F 2.9802322e-8 0)))
(((FAR LEFT) (CLOSE BEHIND) (CLOSE ABOVE) <) ATTACK
 ((K-SL-F -0.45 0) (K-L-F -0.55 0)))
(((FAR LEFT) (CLOSE BEHIND) (CLOSE BELOW) ^) SHOOT-CHASING-OPPONENT
 ((K-L-C 1.25 0) (K-M-C -0.050000012 0) (K-R-C 0.4000002 0)))
(((FAR LEFT) (CLOSE BEHIND) (CLOSE BELOW) >) ATTACK ((K-L-F -0.6999998 0)))
(((FAR LEFT) (CLOSE BEHIND) (CLOSE BELOW) V) ATTACK
 ((K-SL-F 0.15000018 0) (K-L-F -0.29999998 0)))
(((FAR LEFT) (CLOSE BEHIND) (CLOSE BELOW) <) ATTACK
 ((K-SL-F -0.20000003 0) (K-L-F -0.29999995 0)))
(((FAR RIGHT) (CLOSE FRONT) (FAR ABOVE) ^) CATCH-UP
 ((A-M-F -1.0 0) (K-SR-F 0.100000024 0)))
(((FAR RIGHT) (CLOSE FRONT) (FAR ABOVE) >) ATTACK
 ((K-SR-F -0.6 0) (K-R-F -0.29999992 0)))
(((FAR RIGHT) (CLOSE FRONT) (FAR BELOW) >) ATTACK
 ((K-SR-F -0.10000005 0) (K-R-F 0 0)))
(((FAR LEFT) (CLOSE FRONT) (FAR ABOVE) V) AVOID-OPPONENT
 ((A-R-F -0.34999996 0) (K-R-F -0.34999996 0)))
(((FAR LEFT) (CLOSE FRONT) (FAR ABOVE) <) ATTACK
 ((K-SL-F -0.70000017 0) (K-L-F 0 0)))
(((FAR RIGHT) (CLOSE BEHIND) (FAR ABOVE) ^) SHOOT-CHASING-OPPONENT
 ((K-R-C -0.29999995 0) (K-M-C -1.25 0)))
(((FAR RIGHT) (CLOSE BEHIND) (FAR ABOVE) >) ATTACK
```

```
 ((K-SR-F -0.4500001 0) (K-R-F -1.15 0)))
(((FAR RIGHT) (CLOSE BEHIND) (FAR BELOW) >) ATTACK
 ((K-SR-F -1.1999999 0) (K-R-F -1.0 0)))
(((FAR LEFT) (CLOSE BEHIND) (FAR ABOVE) ^) SHOOT-CHASING-OPPONENT
 ((K-M-C -0.8999999 0) (K-R-C 2.9802322e-8 0) (K-L-C -0.0999999 0)))
(((FAR LEFT) (CLOSE BEHIND) (FAR ABOVE) >) ATTACK
 ((K-L-F 0.49999997 0) (K-SL-F 0.050000176 0)))
(((FAR LEFT) (CLOSE BEHIND) (FAR ABOVE) V) ATTACK
 ((K-L-F -1.3999999 0) (K-SL-F -0.4 0)))
(((FAR LEFT) (CLOSE BEHIND) (FAR ABOVE) <) ATTACK
 ((K-L-F -0.100000024 0) (K-SL-F -0.29999998 0)))
(((FAR LEFT) (CLOSE BEHIND) (FAR BELOW) <) ATTACK ((K-SL-F -0.29999998 0)))
(((CLOSE RIGHT) (FAR FRONT) (CLOSE ABOVE) >) ATTACK ((K-R-F 0.45 0)))
(((CLOSE RIGHT) (FAR FRONT) (CLOSE ABOVE) V) AVOID-OPPONENT
 ((A-L-F -1.45 0) (K-L-F 0.30000004 0)))
(((CLOSE RIGHT) (FAR FRONT) (CLOSE ABOVE) <) ATTACK ((K-R-F 0 0)))
(((CLOSE RIGHT) (FAR FRONT) (CLOSE BELOW) >) ATTACK
 ((K-SR-F -0.050000012 0) (K-R-F -1.5 0)))
(((CLOSE RIGHT) (FAR FRONT) (CLOSE BELOW) V) AVOID-OPPONENT
 ((K-L-F 1.4999999 0) (D-L-F 0.4 0) (D-SL-F 0.1 0)))
(((CLOSE RIGHT) (FAR FRONT) (CLOSE BELOW) <) ATTACK
 ((K-R-F 0.7 0) (K-SR-F 0 0)))
(((CLOSE LEFT) (FAR FRONT) (CLOSE ABOVE) ^) CATCH-UP ((K-M-F 0 0)))
(((CLOSE LEFT) (FAR FRONT) (CLOSE ABOVE) >) ATTACK
 ((K-SL-F 0.5000001 0) (K-L-F 0.25000006 0)))
(((CLOSE LEFT) (FAR FRONT) (CLOSE ABOVE) V) AVOID-OPPONENT
 ((K-R-F 0.5999999 0) (A-R-F 0.45000002 0) (A-SR-F 0.35 0)))
(((CLOSE LEFT) (FAR FRONT) (CLOSE ABOVE) <) ATTACK
 ((K-SL-F 0.0 0) (K-L-F -0.45 0)))
(((CLOSE LEFT) (FAR FRONT) (CLOSE BELOW) >) ATTACK
 ((K-SL-F 0.099999994 0) (K-L-F 0.65000004 0)))
(((CLOSE LEFT) (FAR FRONT) (CLOSE BELOW) V) AVOID-OPPONENT
 ((D-R-F 0.44999996 0) (D-SR-F 0.39999992 0) (K-R-F -0.25 0)))
(((CLOSE LEFT) (FAR FRONT) (CLOSE BELOW) <) ATTACK
 ((K-L-F -1.5 0) (K-SL-F -1.1 0)))
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE ABOVE) ^) SHOOT-CHASING-OPPONENT
 ((K-R-C -0.75 0) (K-L-C -0.40000004 0) (K-M-C -0.50000006 0)))
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE ABOVE) >) ATTACK
 ((K-SR-F -1.0999999 0) (K-R-F -1.0999999 0)))
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE ABOVE) V) ATTACK
 ((K-SR-F 0.05000007 0) (K-R-F -0.100000024 0)))
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE ABOVE) <) ATTACK
 ((K-R-F -0.25000006 0) (K-SR-F -0.8999999 0)))
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE BELOW) ^) SHOOT-CHASING-OPPONENT
 ((K-R-C -1.0 0) (K-L-C 0.9999999 0) (K-M-C -0.65000004 0)))
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE BELOW) >) ATTACK
 ((K-SR-F -2.1999998 0) (K-R-F -1.3999999 0)))
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE BELOW) V) ATTACK
 ((K-SR-F -0.65000004 0) (K-R-F -0.100000024 0)))
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE BELOW) <) ATTACK
 ((K-R-F 0.3 0) (K-SR-F -0.4 0)))
(((CLOSE LEFT) (FAR BEHIND) (CLOSE ABOVE) ^) SHOOT-CHASING-OPPONENT
 ((K-R-C -1.25 0) (K-M-C 0.79999995 0) (K-L-C -0.4 0)))
(((CLOSE LEFT) (FAR BEHIND) (CLOSE ABOVE) >) ATTACK
 ((K-L-F 0.04999996 0) (K-SL-F -0.55 0)))
(((CLOSE LEFT) (FAR BEHIND) (CLOSE ABOVE) V) ATTACK
 ((K-SL-F -0.29999998 0) (K-L-F -0.85 0)))
(((CLOSE LEFT) (FAR BEHIND) (CLOSE ABOVE) <) ATTACK
 ((K-L-F -1.1999999 0) (K-SL-F -0.04999996 0)))
(((CLOSE LEFT) (FAR BEHIND) (CLOSE BELOW) ^) SHOOT-CHASING-OPPONENT
 ((K-R-C 0.10000003 0) (K-L-C -3.7252903e-8 0) (K-M-C -2.9802322e-8 0)))
(((CLOSE LEFT) (FAR BEHIND) (CLOSE BELOW) >) ATTACK
 ((K-SL-F -2.1999998 0) (K-L-F -0.30000004 0)))
(((CLOSE LEFT) (FAR BEHIND) (CLOSE BELOW) V) ATTACK
 ((K-L-F 0.10000003 0) (K-SL-F -0.34999996 0)))
(((CLOSE LEFT) (FAR BEHIND) (CLOSE BELOW) <) ATTACK
 ((K-L-F -0.35000002 0) (K-SL-F -0.050000012 0)))
```

```
(((CLOSE RIGHT) (FAR FRONT) (FAR ABOVE) >) ATTACK ((K-R-F -0.9 0)))
(((CLOSE RIGHT) (FAR FRONT) (FAR ABOVE) <) ATTACK ((K-R-F 0 0)))
(((CLOSE LEFT) (FAR FRONT) (FAR ABOVE) <) ATTACK
 ((K-L-F 0 0) (K-SL-F -0.45000005 0)))
(((CLOSE RIGHT) (FAR BEHIND) (FAR ABOVE) ^) SHOOT-CHASING-OPPONENT
 ((K-R-C -0.7 0) (K-L-C -4.0500007 0) (K-M-C -0.40000007 0)))
(((CLOSE RIGHT) (FAR BEHIND) (FAR ABOVE) >) ATTACK ((K-R-F 0.20000005 0)))
(((CLOSE RIGHT) (FAR BEHIND) (FAR ABOVE) V) ATTACK
 ((K-SR-F -0.24999997 0) (K-R-F -1.2999998 0)))
(((CLOSE RIGHT) (FAR BEHIND) (FAR ABOVE) <) ATTACK
 ((K-SR-F -0.79999995 0) (K-R-F 0.24999991 0)))
(((CLOSE LEFT) (FAR BEHIND) (FAR ABOVE) ^) SHOOT-CHASING-OPPONENT
 ((K-L-C -0.54999995 0) (K-R-C -0.85 0) (K-M-C -2.1999998 0)))
(((CLOSE LEFT) (FAR BEHIND) (FAR ABOVE) >) ATTACK
 ((K-L-F -0.05000004 0) (K-SL-F 1.4 0)))
(((CLOSE LEFT) (FAR BEHIND) (FAR ABOVE) <) ATTACK
 ((K-L-F -0.29999998 0) (K-SL-F -0.2 0)))
(((CLOSE LEFT) (FAR BEHIND) (FAR BELOW) <) ATTACK ((K-SL-F 0 0)))
(((FAR LEFT) (FAR FRONT) (CLOSE ABOVE) ^) CATCH-UP ((K-SR-F 0 0)))
(((FAR LEFT) (FAR FRONT) (CLOSE BELOW) <) ATTACK
 ((K-L-F 0.34999996 0) (K-SL-F -0.39999998 0)))
(((FAR RIGHT) (FAR BEHIND) (CLOSE ABOVE) ^) SHOOT-CHASING-OPPONENT
 ((K-R-C -0.19999999 0) (K-L-C -0.09999998 0) (K-M-C -0.14999998 0)))
(((FAR RIGHT) (FAR BEHIND) (CLOSE ABOVE) >) ATTACK
 ((K-SR-F -0.65 0) (K-R-F -0.09999998 0)))
(((FAR RIGHT) (FAR BEHIND) (CLOSE ABOVE) V) ATTACK
 ((K-SR-F 0.049999923 0) (K-R-F -0.35000008 0)))
(((FAR RIGHT) (FAR BEHIND) (CLOSE ABOVE) <) ATTACK
 ((K-SR-F -0.29999995 0) (K-R-F -0.25000006 0)))
(((FAR RIGHT) (FAR BEHIND) (CLOSE BELOW) ^) SHOOT-CHASING-OPPONENT
 ((K-M-C -0.09999998 0) (K-R-C 0.10000004 0) (K-L-C -0.30000007 0)))
(((FAR RIGHT) (FAR BEHIND) (CLOSE BELOW) >) ATTACK
 ((K-SR-F -0.20000003 0) (K-R-F -0.09999998 0)))
(((FAR RIGHT) (FAR BEHIND) (CLOSE BELOW) V) ATTACK
 ((K-SR-F -0.45000005 0) (K-R-F -0.25000006 0)))
(((FAR RIGHT) (FAR BEHIND) (CLOSE BELOW) <) ATTACK
 ((K-SR-F 0.05000004 0) (K-R-F -0.30000004 0)))
(((FAR LEFT) (FAR BEHIND) (CLOSE ABOVE) ^) SHOOT-CHASING-OPPONENT
 ((K-L-C 5.9604645e-8 0) (K-M-C -0.20000003 0) (K-R-C -0.09999998 0)))
(((FAR LEFT) (FAR BEHIND) (CLOSE ABOVE) >) ATTACK
 ((K-SL-F 5.9604645e-8 0) (K-L-F -0.25000006 0)))
(((FAR LEFT) (FAR BEHIND) (CLOSE ABOVE) V) ATTACK
 ((K-L-F -0.20000003 0) (K-SL-F -2.2351742e-8 0)))
(((FAR LEFT) (FAR BEHIND) (CLOSE ABOVE) <) ATTACK
 ((K-SL-F -0.45 0) (K-L-F -0.100000024 0)))
(((FAR LEFT) (FAR BEHIND) (CLOSE BELOW) ^) SHOOT-CHASING-OPPONENT
 ((K-R-C -0.09999998 0) (K-M-C -0.25000006 0) (K-L-C -0.25000006 0)))
(((FAR LEFT) (FAR BEHIND) (CLOSE BELOW) >) ATTACK
 ((K-L-F 2.9802322e-8 0) (K-SL-F 0.20000005 0)))
(((FAR LEFT) (FAR BEHIND) (CLOSE BELOW) V) ATTACK
 ((K-SL-F -0.30000004 0) (K-L-F -0.40000004 0)))
(((FAR LEFT) (FAR BEHIND) (CLOSE BELOW) <) ATTACK
 ((K-SL-F -0.20000003 0) (K-L-F -0.100000024 0)))
(((FAR RIGHT) (FAR BEHIND) (FAR ABOVE) >) ATTACK
 ((K-SR-F -0.19999999 0) (K-R-F -0.34999996 0)))
(((FAR LEFT) (FAR BEHIND) (FAR ABOVE) ^) SHOOT-CHASING-OPPONENT
 ((K-M-C -0.100000024 0) (K-L-C -0.24999993 0) (K-R-C -0.20000003 0)))
(((FAR LEFT) (FAR BEHIND) (FAR ABOVE) >) ATTACK
 ((K-SL-F 0.050000057 0) (K-L-F -0.24999999 0)))
(((FAR LEFT) (FAR BEHIND) (FAR ABOVE) V) ATTACK
 ((K-L-F -0.40000004 0) (K-SL-F -0.35 0)))
(((FAR LEFT) (FAR BEHIND) (FAR ABOVE) <) ATTACK
 ((K-SL-F -0.29999998 0) (K-L-F -0.100000024 0)))
(((FAR LEFT) (FAR BEHIND) (FAR BELOW) ^) SHOOT-CHASING-OPPONENT
 ((K-R-C -0.40000004 0) (K-M-C -0.100000024 0) (K-L-C 0.050000057 0)))
)
```

A.2 Behavior Transitions

The following is a list of Gabby's behavior transitions after 500 games. Each transition is a triple of Situation (e.g.,((FAR LEFT) (FAR FRONT) (CLOSE ABOVE) <)) , Old behavior (e.g., ATTACK) , New behavior (e.g., CATCH-UP).

```
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE BELOW) ^) ASSUME-ATTACK-POSITION
 MAINTAIN-ATTACKING)
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE BELOW) ^) CATCH-UP MAINTAIN-ATTACKING)
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE BELOW) ^) ATTACK MAINTAIN-ATTACKING)
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE BELOW) >) MAINTAIN-ATTACKING
 ASSUME-ATTACK-POSITION)
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE BELOW) >) AVOID-ATTACK
 ASSUME-ATTACK-POSITION)
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE BELOW) >) ATTACK ASSUME-ATTACK-POSITION)
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE BELOW) V) ATTACK AVOID-CRASH)
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE BELOW) <) MAINTAIN-ATTACKING
 ASSUME-ATTACK-POSITION)
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE BELOW) <) AVOID-ATTACK
 ASSUME-ATTACK-POSITION)
(((CLOSE LEFT) (CLOSE FRONT) (CLOSE BELOW) <) ATTACK ASSUME-ATTACK-POSITION)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) ^) AVOID-CRASH AVOID-ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) ^) ASSUME-ATTACK-POSITION
 AVOID-ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) ^) CATCH-UP AVOID-ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) ^) ATTACK AVOID-ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) ^) SHOOT-CHASING-OPPONENT
 AVOID-ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) >) MAINTAIN-ATTACKING ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) >) ASSUME-ATTACK-POSITION ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) >) AVOID-ATTACK ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) >) AVOID-OPPONENT ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) >) CATCH-UP ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) V) MAINTAIN-ATTACKING ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) V) ASSUME-ATTACK-POSITION ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) V) AVOID-ATTACK ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) V) AVOID-OPPONENT ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) V) CATCH-UP ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) V) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) <) MAINTAIN-ATTACKING ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) <) ASSUME-ATTACK-POSITION ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) <) AVOID-ATTACK ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) <) CATCH-UP ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) <) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) ^) AVOID-CRASH AVOID-ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) ^) ASSUME-ATTACK-POSITION
 AVOID-ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) ^) CATCH-UP AVOID-ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) ^) ATTACK AVOID-ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) ^) SHOOT-CHASING-OPPONENT
 AVOID-ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) >) MAINTAIN-ATTACKING ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) >) ASSUME-ATTACK-POSITION ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) >) AVOID-ATTACK ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) >) AVOID-OPPONENT ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) >) CATCH-UP ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) >) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) V) ASSUME-ATTACK-POSITION ATTACK)
```

```
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) V) AVOID-ATTACK ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) V) AVOID-OPPONENT ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) V) CATCH-UP ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) V) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) <) MAINTAIN-ATTACKING ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) <) ASSUME-ATTACK-POSITION ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) <) AVOID-ATTACK ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) <) AVOID-OPPONENT ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) <) CATCH-UP ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (CLOSE BELOW) <) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) ^) AVOID-CRASH AVOID-ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) ^) ASSUME-ATTACK-POSITION
 AVOID-ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) ^) ATTACK AVOID-ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) ^) SHOOT-CHASING-OPPONENT
 AVOID-ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) >) MAINTAIN-ATTACKING ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) >) ASSUME-ATTACK-POSITION ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) >) AVOID-ATTACK ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) >) CATCH-UP ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) V) MAINTAIN-ATTACKING ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) V) AVOID-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) V) CATCH-UP ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) V) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) <) MAINTAIN-ATTACKING ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) <) ASSUME-ATTACK-POSITION ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) <) AVOID-ATTACK ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) <) AVOID-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) <) CATCH-UP ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE ABOVE) <) SHOOT-CHASIPPNG-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) ^) AVOID-CRASH AVOID-ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) ^) ASSUME-ATTACK-POSITION
 AVOID-ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) ^) CATCH-UP AVOID-ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) ^) ATTACK AVOID-ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) ^) SHOOT-CHASING-OPPONENT
 AVOID-ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) >) MAINTAIN-ATTACKING ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) >) ASSUME-ATTACK-POSITION ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) >) AVOID-ATTACK ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) >) CATCH-UP ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) >) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) V) MAINTAIN-ATTACKING ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) V) AVOID-CRASH ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) V) AVOID-ATTACK ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) V) AVOID-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) V) CATCH-UP ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) V) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) <) MAINTAIN-ATTACKING ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) <) ASSUME-ATTACK-POSITION ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) <) AVOID-ATTACK ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) <) AVOID-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) <) CATCH-UP ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (CLOSE BELOW) <) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (CLOSE FRONT) (FAR ABOVE) ^) MAINTAIN-ATTACKING CATCH-UP)
(((CLOSE RIGHT) (CLOSE FRONT) (FAR ABOVE) >) MAINTAIN-ATTACKING ATTACK)
(((CLOSE RIGHT) (CLOSE FRONT) (FAR ABOVE) >) AVOID-ATTACK ATTACK)
(((CLOSE RIGHT) (CLOSE FRONT) (FAR ABOVE) <) MAINTAIN-ATTACKING ATTACK)
(((CLOSE RIGHT) (CLOSE FRONT) (FAR ABOVE) <) AVOID-ATTACK ATTACK)
(((CLOSE RIGHT) (CLOSE FRONT) (FAR BELOW) >) AVOID-ATTACK ATTACK)
(((CLOSE RIGHT) (CLOSE FRONT) (FAR BELOW) <) MAINTAIN-ATTACKING ATTACK)
(((CLOSE RIGHT) (CLOSE FRONT) (FAR BELOW) <) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE FRONT) (FAR ABOVE) ^) MAINTAIN-ATTACKING CATCH-UP)
(((CLOSE LEFT) (CLOSE FRONT) (FAR ABOVE) >) AVOID-ATTACK ATTACK)
(((CLOSE LEFT) (CLOSE FRONT) (FAR ABOVE) <) MAINTAIN-ATTACKING ATTACK)
(((CLOSE LEFT) (CLOSE FRONT) (FAR ABOVE) <) AVOID-ATTACK ATTACK)
```

```
(((CLOSE LEFT) (CLOSE FRONT) (FAR BELOW) ^) MAINTAIN-ATTACKING CATCH-UP)
(((CLOSE LEFT) (CLOSE FRONT) (FAR BELOW) >) AVOID-ATTACK ATTACK)
(((CLOSE LEFT) (CLOSE FRONT) (FAR BELOW) <) MAINTAIN-ATTACKING ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR ABOVE) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR ABOVE) >) MAINTAIN-ATTACKING ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR ABOVE) >) AVOID-ATTACK ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR ABOVE) >) AVOID-OPPONENT ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR ABOVE) V) AVOID-OPPONENT ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR ABOVE) V) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR ABOVE) <) MAINTAIN-ATTACKING ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR ABOVE) <) AVOID-ATTACK ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR ABOVE) <) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR BELOW) >) AVOID-ATTACK ATTACK)
(((CLOSE RIGHT) (CLOSE BEHIND) (FAR BELOW) <) AVOID-ATTACK ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (FAR ABOVE) ^) AVOID-OPPONENT
 SHOOT-CHASING-OPPONENT)
(((CLOSE LEFT) (CLOSE BEHIND) (FAR ABOVE) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((CLOSE LEFT) (CLOSE BEHIND) (FAR ABOVE) >) AVOID-ATTACK ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (FAR ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (FAR ABOVE) V) AVOID-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (FAR ABOVE) V) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (FAR ABOVE) <) AVOID-ATTACK ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (FAR ABOVE) <) AVOID-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (FAR ABOVE) <) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (FAR BELOW) >) AVOID-ATTACK ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (FAR BELOW) V) AVOID-OPPONENT ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (FAR BELOW) <) AVOID-ATTACK ATTACK)
(((CLOSE LEFT) (CLOSE BEHIND) (FAR BELOW) <) CATCH-UP ATTACK)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE ABOVE) ^) ASSUME-ATTACK-POSITION CATCH-UP)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE ABOVE) ^) AVOID-OPPONENT CATCH-UP)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE ABOVE) ^) ATTACK CATCH-UP)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE ABOVE) >) MAINTAIN-ATTACKING ATTACK)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE ABOVE) >) ASSUME-ATTACK-POSITION ATTACK)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE ABOVE) >) AVOID-ATTACK ATTACK)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE ABOVE) >) AVOID-OPPONENT ATTACK)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE ABOVE) >) CATCH-UP ATTACK)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE ABOVE) V) ATTACK AVOID-OPPONENT)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE BELOW) ^) ASSUME-ATTACK-POSITION CATCH-UP)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE BELOW) ^) AVOID-OPPONENT CATCH-UP)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE BELOW) ^) ATTACK CATCH-UP)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE BELOW) >) ASSUME-ATTACK-POSITION ATTACK)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE BELOW) >) AVOID-ATTACK ATTACK)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE BELOW) >) AVOID-OPPONENT ATTACK)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE BELOW) >) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE BELOW) V) ASSUME-ATTACK-POSITION
 AVOID-OPPONENT)
(((FAR RIGHT) (CLOSE FRONT) (CLOSE BELOW) V) ATTACK AVOID-OPPONENT)
(((FAR LEFT) (CLOSE FRONT) (CLOSE ABOVE) ^) MAINTAIN-ATTACKING CATCH-UP)
(((FAR LEFT) (CLOSE FRONT) (CLOSE ABOVE) ^) AVOID-OPPONENT CATCH-UP)
(((FAR LEFT) (CLOSE FRONT) (CLOSE ABOVE) V) ATTACK AVOID-OPPONENT)
(((FAR LEFT) (CLOSE FRONT) (CLOSE ABOVE) <) MAINTAIN-ATTACKING ATTACK)
(((FAR LEFT) (CLOSE FRONT) (CLOSE ABOVE) <) CATCH-UP ATTACK)
(((FAR LEFT) (CLOSE FRONT) (CLOSE ABOVE) <) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (CLOSE FRONT) (CLOSE BELOW) ^) MAINTAIN-ATTACKING CATCH-UP)
(((FAR LEFT) (CLOSE FRONT) (CLOSE BELOW) ^) AVOID-OPPONENT CATCH-UP)
(((FAR LEFT) (CLOSE FRONT) (CLOSE BELOW) ^) ATTACK CATCH-UP)
(((FAR LEFT) (CLOSE FRONT) (CLOSE BELOW) >) MAINTAIN-ATTACKING ATTACK)
(((FAR LEFT) (CLOSE FRONT) (CLOSE BELOW) >) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (CLOSE FRONT) (CLOSE BELOW) V) ATTACK AVOID-OPPONENT)
(((FAR LEFT) (CLOSE FRONT) (CLOSE BELOW) <) MAINTAIN-ATTACKING ATTACK)
(((FAR LEFT) (CLOSE FRONT) (CLOSE BELOW) <) ASSUME-ATTACK-POSITION ATTACK)
(((FAR LEFT) (CLOSE FRONT) (CLOSE BELOW) <) AVOID-OPPONENT ATTACK)
(((FAR LEFT) (CLOSE FRONT) (CLOSE BELOW) <) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) ^) AVOID-OPPONENT
 SHOOT-CHASING-OPPONENT)
```

```
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) >) ASSUME-ATTACK-POSITION ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) >) AVOID-ATTACK ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) >) CATCH-UP ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) V) AVOID-ATTACK ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) V) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE ABOVE) <) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE BELOW) ^) ASSUME-ATTACK-POSITION
 SHOOT-CHASING-OPPONENT)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE BELOW) ^) AVOID-OPPONENT
 SHOOT-CHASING-OPPONENT)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE BELOW) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE BELOW) >) ASSUME-ATTACK-POSITION ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE BELOW) >) AVOID-ATTACK ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE BELOW) >) AVOID-OPPONENT ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE BELOW) >) CATCH-UP ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE BELOW) >) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE BELOW) V) ASSUME-ATTACK-POSITION ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE BELOW) V) CATCH-UP ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE BELOW) V) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (CLOSE BELOW) <) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (CLOSE BEHIND) (CLOSE ABOVE) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((FAR LEFT) (CLOSE BEHIND) (CLOSE ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (CLOSE BEHIND) (CLOSE ABOVE) V) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (CLOSE BEHIND) (CLOSE ABOVE) <) AVOID-ATTACK ATTACK)
(((FAR LEFT) (CLOSE BEHIND) (CLOSE ABOVE) <) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (CLOSE BEHIND) (CLOSE BELOW) ^) AVOID-OPPONENT
 SHOOT-CHASING-OPPONENT)
(((FAR LEFT) (CLOSE BEHIND) (CLOSE BELOW) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((FAR LEFT) (CLOSE BEHIND) (CLOSE BELOW) >) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (CLOSE BEHIND) (CLOSE BELOW) V) CATCH-UP ATTACK)
(((FAR LEFT) (CLOSE BEHIND) (CLOSE BELOW) V) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (CLOSE BEHIND) (CLOSE BELOW) <) AVOID-ATTACK ATTACK)
(((FAR LEFT) (CLOSE BEHIND) (CLOSE BELOW) <) CATCH-UP ATTACK)
(((FAR LEFT) (CLOSE BEHIND) (CLOSE BELOW) <) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR RIGHT) (CLOSE FRONT) (FAR ABOVE) ^) AVOID-OPPONENT CATCH-UP)
(((FAR RIGHT) (CLOSE FRONT) (FAR ABOVE) ^) ATTACK CATCH-UP)
(((FAR RIGHT) (CLOSE FRONT) (FAR ABOVE) >) AVOID-ATTACK ATTACK)
(((FAR RIGHT) (CLOSE FRONT) (FAR ABOVE) >) AVOID-OPPONENT ATTACK)
(((FAR RIGHT) (CLOSE FRONT) (FAR BELOW) >) AVOID-ATTACK ATTACK)
(((FAR LEFT) (CLOSE FRONT) (FAR ABOVE) V) ATTACK AVOID-OPPONENT)
(((FAR LEFT) (CLOSE FRONT) (FAR ABOVE) <) AVOID-OPPONENT ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (FAR ABOVE) ^) AVOID-OPPONENT
 SHOOT-CHASING-OPPONENT)
(((FAR RIGHT) (CLOSE BEHIND) (FAR ABOVE) >) MAINTAIN-ATTACKING ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (FAR ABOVE) >) AVOID-ATTACK ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (FAR ABOVE) >) AVOID-OPPONENT ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (FAR ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR RIGHT) (CLOSE BEHIND) (FAR BELOW) >) AVOID-ATTACK ATTACK)
(((FAR LEFT) (CLOSE BEHIND) (FAR ABOVE) ^) AVOID-OPPONENT
 SHOOT-CHASING-OPPONENT)
(((FAR LEFT) (CLOSE BEHIND) (FAR ABOVE) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((FAR LEFT) (CLOSE BEHIND) (FAR ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (CLOSE BEHIND) (FAR ABOVE) V) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (CLOSE BEHIND) (FAR ABOVE) <) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (CLOSE BEHIND) (FAR BELOW) <) AVOID-ATTACK ATTACK)
(((CLOSE RIGHT) (FAR FRONT) (CLOSE ABOVE) >) ASSUME-ATTACK-POSITION ATTACK)
(((CLOSE RIGHT) (FAR FRONT) (CLOSE ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (FAR FRONT) (CLOSE ABOVE) V) AVOID-CRASH AVOID-OPPONENT)
(((CLOSE RIGHT) (FAR FRONT) (CLOSE ABOVE) <) CATCH-UP ATTACK)
(((CLOSE RIGHT) (FAR FRONT) (CLOSE BELOW) >) ASSUME-ATTACK-POSITION ATTACK)
(((CLOSE RIGHT) (FAR FRONT) (CLOSE BELOW) >) AVOID-ATTACK ATTACK)
(((CLOSE RIGHT) (FAR FRONT) (CLOSE BELOW) >) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (FAR FRONT) (CLOSE BELOW) V) AVOID-CRASHG AVOID-OPPONENT)
(((CLOSE RIGHT) (FAR FRONT) (CLOSE BELOW) V) ATTACK AVOID-OPPONENT)
(((CLOSE RIGHT) (FAR FRONT) (CLOSE BELOW) <) SHOOT-CHASING-OPPONENT ATTACK)
```

```
(((CLOSE LEFT) (FAR FRONT) (CLOSE ABOVE) ^) MAINTAIN-ATTACKING CATCH-UP)
(((CLOSE LEFT) (FAR FRONT) (CLOSE ABOVE) >) MAINTAIN-ATTACKING ATTACK)
(((CLOSE LEFT) (FAR FRONT) (CLOSE ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (FAR FRONT) (CLOSE ABOVE) V) AVOID-CRASH AVOID-OPPONENT)
(((CLOSE LEFT) (FAR FRONT) (CLOSE ABOVE) V) ATTACK AVOID-OPPONENT)
(((CLOSE LEFT) (FAR FRONT) (CLOSE ABOVE) <) MAINTAIN-ATTACKING ATTACK)
(((CLOSE LEFT) (FAR FRONT) (CLOSE ABOVE) <) ASSUME-ATTACK-POSITION ATTACK)
(((CLOSE LEFT) (FAR FRONT) (CLOSE ABOVE) <) AVOID-ATTACK ATTACK)
(((CLOSE LEFT) (FAR FRONT) (CLOSE BELOW) >) MAINTAIN-ATTACKING ATTACK)
(((CLOSE LEFT) (FAR FRONT) (CLOSE BELOW) >) AVOID-ATTACK ATTACK)
(((CLOSE LEFT) (FAR FRONT) (CLOSE BELOW) >) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (FAR FRONT) (CLOSE BELOW) V) AVOID-CRASH AVOID-OPPONENT)
(((CLOSE LEFT) (FAR FRONT) (CLOSE BELOW) V) ATTACK AVOID-OPPONENT)
(((CLOSE LEFT) (FAR FRONT) (CLOSE BELOW) <) MAINTAIN-ATTACKING ATTACK)
(((CLOSE LEFT) (FAR FRONT) (CLOSE BELOW) <) ASSUME-ATTACK-POSITION ATTACK)
(((CLOSE LEFT) (FAR FRONT) (CLOSE BELOW) <) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE ABOVE) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE ABOVE) >) AVOID-ATTACK ATTACK)
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE ABOVE) V) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE ABOVE) <) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE BELOW) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE BELOW) >) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE BELOW) V) CATCH-UP ATTACK)
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE BELOW) V) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE BELOW) <) AVOID-OPPONENT ATTACK)
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE BELOW) <) CATCH-UP ATTACK)
(((CLOSE RIGHT) (FAR BEHIND) (CLOSE BELOW) <) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE ABOVE) ^) AVOID-OPPONENT
 SHOOT-CHASING-OPPONENT)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE ABOVE) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE ABOVE) >) MAINTAIN-ATTACKING ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE ABOVE) >) CATCH-UP ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE ABOVE) V) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE ABOVE) <) AVOID-OPPONENT ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE ABOVE) <) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE BELOW) ^) AVOID-OPPONENT
 SHOOT-CHASING-OPPONENT)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE BELOW) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE BELOW) >) MAINTAIN-ATTACKING ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE BELOW) >) AVOID-ATTACK ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE BELOW) >) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE BELOW) V) CATCH-UP ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE BELOW) V) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE BELOW) <) AVOID-OPPONENT ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE BELOW) <) CATCH-UP ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (CLOSE BELOW) <) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (FAR FRONT) (FAR ABOVE) >) MAINTAIN-ATTACKING ATTACK)
(((CLOSE RIGHT) (FAR FRONT) (FAR ABOVE) <) MAINTAIN-ATTACKING ATTACK)
(((CLOSE LEFT) (FAR FRONT) (FAR ABOVE) <) MAINTAIN-ATTACKING ATTACK)
(((CLOSE RIGHT) (FAR BEHIND) (FAR ABOVE) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((CLOSE RIGHT) (FAR BEHIND) (FAR ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (FAR BEHIND) (FAR ABOVE) V) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE RIGHT) (FAR BEHIND) (FAR ABOVE) <) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (FAR ABOVE) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((CLOSE LEFT) (FAR BEHIND) (FAR ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (FAR ABOVE) <) SHOOT-CHASING-OPPONENT ATTACK)
(((CLOSE LEFT) (FAR BEHIND) (FAR BELOW) <) CATCH-UP ATTACK)
(((FAR LEFT) (FAR FRONT) (CLOSE ABOVE) ^) MAINTAIN-ATTACKING CATCH-UP)
(((FAR LEFT) (FAR FRONT) (CLOSE BELOW) <) MAINTAIN-ATTACKING ATTACK)
(((FAR RIGHT) (FAR BEHIND) (CLOSE ABOVE) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((FAR RIGHT) (FAR BEHIND) (CLOSE ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR RIGHT) (FAR BEHIND) (CLOSE ABOVE) V) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR RIGHT) (FAR BEHIND) (CLOSE ABOVE) <) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR RIGHT) (FAR BEHIND) (CLOSE BELOW) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((FAR RIGHT) (FAR BEHIND) (CLOSE BELOW) ^) NIL SHOOT-CHASING-OPPONENT)
```

```
(((FAR RIGHT) (FAR BEHIND) (CLOSE BELOW) >) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR RIGHT) (FAR BEHIND) (CLOSE BELOW) V) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR RIGHT) (FAR BEHIND) (CLOSE BELOW) <) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (FAR BEHIND) (CLOSE ABOVE) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((FAR LEFT) (FAR BEHIND) (CLOSE ABOVE) ^) NIL SHOOT-CHASING-OPPONENT)
(((FAR LEFT) (FAR BEHIND) (CLOSE ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (FAR BEHIND) (CLOSE ABOVE) V) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (FAR BEHIND) (CLOSE ABOVE) <) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (FAR BEHIND) (CLOSE BELOW) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((FAR LEFT) (FAR BEHIND) (CLOSE BELOW) ^) NIL SHOOT-CHASING-OPPONENT)
(((FAR LEFT) (FAR BEHIND) (CLOSE BELOW) >) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (FAR BEHIND) (CLOSE BELOW) V) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (FAR BEHIND) (CLOSE BELOW) <) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR RIGHT) (FAR BEHIND) (FAR ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (FAR BEHIND) (FAR ABOVE) ^) ATTACK SHOOT-CHASING-OPPONENT)
(((FAR LEFT) (FAR BEHIND) (FAR ABOVE) ^) NIL SHOOT-CHASING-OPPONENT)
(((FAR LEFT) (FAR BEHIND) (FAR ABOVE) >) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (FAR BEHIND) (FAR ABOVE) V) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (FAR BEHIND) (FAR ABOVE) <) SHOOT-CHASING-OPPONENT ATTACK)
(((FAR LEFT) (FAR BEHIND) (FAR BELOW) ^) NIL SHOOT-CHASING-OPPONENT)
)
```

Appendix B

Notes

This Appendix contains a few defiinitions and a few useful quotations about terms used in this dissertation.

- **Artificial Life**– "Artificial Life is the study of man-made systems that exhibit behaviors characteristic of natural living systems. It complements the traditional biological sciences concerned with the analysis of living organisms by attempting to synthesize life-like behaviors within computers and other artificial media. By extending the empirical foundation upon which biology is based beyond the carbon-chain life that has evolved on Earth. Artificial Life can contribute to theoretical biology by locating life-as-we-know-it within the larger picture of life-as-it-could-be." [Langton, 1989, p. 1].

- **Action**– "the smallest unit of behavior that is related to a conscious goal." [Frese and Zapf, 1994, p. 274]

- **Action slips**– "They often appear to result from conflict among several possible actions or thoughts, from intermixing the components of a single action sequence, or from selection of an appropriate act but in some inappropriate way." [Norman, 1981]

- **Action Schemata**– Parameterized plans for motor actions. [Schmidt, 1975, Norman, 1981]

- **Behavior**– Unintended action. "Acts as I use the term, are what humans do when they intend to do something, whereas behavior is unintended." [Collins, 1990, p. 30].

- **Consciousness**– We define consciousness for a robotic agent operationally as being aware of its environment, as evidenced by (1) having some internal states or representations that are causally connected to the environment through perception, (2) being able to reason explicitly about the environment, (3) being able to communicate with an external agent about the environment, and (4) having internal states of a sense of self.

  Simply being able to verbalize a thought is not a defining criteria.[Shepard and Metzler, 1971]. Also, having a response for stimulus is not adequate for consciousness [Culbertson, 1963]. A vending machine or an industrial robot has responses, but it is not conscious of the interaction.

- **Classical Conditioning**– Pair the stimulus to be conditioned (NS) with a stimulus (US) that elicits a response (UR). Eventually, the NS elicits the response (CR), so NS becomes CS.

- **Error**– "A)Errors only appear in goal-oriented action, B) They imply the nonattainment of goals, C) an error should have been potentially avoidable" [Frese and Zapf, 1994, p. 288].

- **Goal Sketch**– "A goal sketch specifies the hierarchy of objectives that a satisfactory strategy must meet." [Siegler and Jenkins, 1989, p. 115]

- **Habit**– "By a habit it is generally understood a stereotyped form of response: the doing of the same thing always in the same way under the same condition".[Dunlap, 1932, p. 8]

- **Knowledge**– Part of an agent's mental/physical abilities that can be modeled with a strong symbol system.

- **Markovian assumption**– The Markovian property is that the current state (no memory) is all that is needed to produce an optimal action policy. The Markovian assumption for an agent is that the effect of an agent's actions only depends on its current state.

- **Microgenetic method**– Key properties from [Siegler and Jenkins, 1989, p. 9]: "(a) Subjects are observed over an extended period of time, (b) Their learning is subjected to intensive trial-by-trial analysis, with the goal being to infer the underlying representations and processes that gave rise to both quantitative and qualitative aspects of learning".

- **Personalized act**– A learned combination of similar or dissimilar primitive acts, which at times coincides with a *skill* or a *habit*.

- **Operant Conditioning**– Present positive reinforcer or remove a negative reinforcer immediately after an organism makes a response. In contrast to classical conditioning, this involves voluntary responses.

- **Plan**– "Behavior that is voluntary, self-conscious, and intended" [Scholnick and Friedman, 1987, p. 5].

- **Redundant environment**– An environment in which there is more than one way of accomplishing a task.

- **Routine**– A course of action that is habitually (or by choice) adhered to.

- **Skill**– Part of an agent's mental/physical abilities that can't be modeled with a strong symbol system.

- **Strategy**– "Any procedure that is non-obligatory and goal directed ... pulling back the gear shift on a car to move from first gear to second is a procedure and not a strategy, because there is no alternative way to shift. ... strategies necessarily involve choice" [Siegler and Jenkins, 1989, p. 11 and 12].

- **Structural credit assignment**– how to propagate the reward spatially across states so that similar states cause the agent to take similar actions.

- **Temporal credit assignment**– how to propagate the reward backwards in time.

Bibliography

[Adams, 1971] Adams, J. (1971). A closed-loop theory of motor learning. *Motor Behavior*, 3:111–149.

[Agre, 1988] Agre, P. (1988). The dynamic structure of everyday life. Technical Report 1085, MIT Artificial Intelligence Laboratory, MIT.

[Agre and Chapman, 1990] Agre, P. and Chapman, D. (1990). What are plans for? In Maes, P., editor, *Designing Autonomous Agents*, chapter 2, pages 17–34. MIT Press.

[Agre and Chapman, 1987] Agre, P. E. and Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *Proceedings of AAAI-87, Seattle, Wa.*, pages 268–272.

[Agre and Shrager, 1990] Agre, P. E. and Shrager, J. (1990). Routine evolution as the microgenetic basis of skill acquisition. In *Proceedings Cognitive Science Conference- Boston*, pages 694–701.

[Albus, 1995] Albus, J. (1995). RCS: a reference model architecture for intelligent systems. In *AAAI Spring Symposium on Lessons Learned From Implemented Software Architectures for Physical Agents*, pages 1–6. AAAI press.

[Albus et al., 1981] Albus, J., Barbera, A., and Nagel, R. (1981). Theory and practice of hierarchical control. In *23rd International IEEE Computer Society Conference*, pages 18–38.

[Alterman, 1986] Alterman, R. (1986). An adaptive planner. In *Proceedings of AAAI-86*, pages 65–69, Boston.

[Anderson, 1993] Anderson, J. R. (1993). *Rules of the Mind*. Lawrence Earlbaum, Hillsdale, NJ.

[Bjorklund, 1989] Bjorklund, D. (1989). *Children's Thinking*. Brooks/Cole, Pacific Grove, CA.

[Blythe and Mitchell, 1989] Blythe, J. and Mitchell, T. (1989). On becoming reactive. In *Machine Learning Workshop*, pages 255–257.

[Bonasso and Kortenkamp, 1995] Bonasso, R. P. and Kortenkamp, D. (1995). Characterizing an architecture for intelligent, reactive agents. In *AAAI Spring Symposium on Lessons Learned From Implemented Software Architectures for Physical Agents*, pages 1–6. AAAI press.

[Brady and Hu, 1995] Brady, M. and Hu, H. (1995). Software and hardware architceture of a mobile robot for manufacturing. In *AAAI Spring Symposium on Lessons Learned From Implemented Software Architectures for Physical Agents*, pages 1–6. AAAI press.

[Brooks, 1985] Brooks, R. (1985). A robust layered control system for a mobile robot. Technical Report 864, MIT AI Labs, MIT.

[Brooks, 1987] Brooks, R. (1987). Planning is just a way of avoiding figuring out what to do next. Technical Report 303, MIT AI Labs.

[Brooks, 1990] Brooks, R. A. (1990). Elephants dont play chess. In Maes, P., editor, *Designing Autonomous Agents*, pages 3–15. MIT Press.

[Caicedo and Hexmoor, 1993] Caicedo, G. and Hexmoor, H. (1993). Alignment in GLAIR. Technical Report (Unpublished), Computer Science Department, State University of New York at Buffalo, Buffalo, NY.

[Card et al., 1983] Card, S., Moran, T., and Newell, A. (1983). *The Psychology of Human-Computer Interaction.* Erlbaum, Hillsdale, N.J.

[Chapman, 1990] Chapman, D. (1990). Vision, instruction, and action. Technical Report 1204, MIT Artificial Intelligence Laboratory, MIT.

[Christiansen et al., 1990] Christiansen, A., Mason, M., and Mitchell, T. (1990). Learning reliable manipulation strategies without initial physical models. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 1224–1230.

[Collins, 1990] Collins, H. (1990). *Artificial Experts: Social Knowledge and Intelligent Machines.* The MIT Press, Cambridge, Massachusetts.

[Connell, 1992] Connell, J. (1992). SSS: a hybrid architecture applied to robot navigation. In *IEEE Conference on Robotics and Automation*, pages 2719–2724.

[Culbertson, 1963] Culbertson, J. (1963). *The Minds of Robots.* U. of Illinois Press.

[DeJong and Mooney, 1986] DeJong, G. and Mooney, R. (1986). Explanation-based learning: An alternative view. In *Machine Learning 1*, pages 144–176. Kluwer Academic Publishers, Boston.

[Drummond, 1989] Drummond, M. (1989). Situated control rules. In *Proceedings Of the first conference on principles of knowledge representation*, pages 103–113. Morgan Kaufman.

[Dunlap, 1932] Dunlap, K. (1932). *Habits, their making and unmaking.* Liveright, New York, NY.

[Dunnette, 1983] Dunnette, M., editor (1983). *Handbook of Industrial and Organizational Psychology.* John Wiley and Sons.

[Fikes et al., 1972] Fikes, R., Hart, P., and Nilsson, N. (1972). Learning and executing generalized robot plans. In *Artificial Intelligence 3*, pages 231–249. Elsevier Science Publishers.

[Fikes and Nilsson, 1971] Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 5:189–208.

[Firby, 1987] Firby, R. J. (1987). An investigation into reactive planning in complex domains. In *Proceedings of AAAI-87*, pages 202–206, Seattle, WA.

[Fodor, 1983] Fodor, J. (1983). *The Modularity of Mind.* MIT Press.

[Frese and Zapf, 1994] Frese, M. and Zapf, D. (1994). Action as the core of work psychology:a german approach. In Triaandis, H., Dunette, M., and Hough, L., editors, *Handbook of Industrial and Organizational Psychology*, chapter 6, pages 271–340. Consulting Psychlogists Press, Palo Alto, California.

[Gat, 1991] Gat, E. (1991). Reliable goal-directed reactive control of autonomous mobile robot. Technical Report Tec, Dept. of Computer Science, Virginia Polytechnic Institute and State University.

[Gervasio, 1990] Gervasio, M. (1990). Learning general completable reactive plans. In *Proceedings of AAAI-90*, pages 1016–1021.

[Gervasio and DeJong, 1989] Gervasio, M. and DeJong, G. (1989). Explanation-based learning of reactive operators. In *Machine Learning Workshop*, pages 252– 254.

[Hammond, 1986] Hammond, K. J. (1986). Case-based planning: An integrated theory of planning, learning, and memory. Technical Report YALEEU/CSD/RR 488, Dept. of Computer Science, Yale University.

[Harnad, 1990] Harnad, S. (1990). The symbol grounding problem. *Physica D*, 42(1-3):335–346.

[Harnad, 1992] Harnad, S. (1992). Electronic symposium on computation, cognition and the symbol grounding problem. E-mail symposium (ftp archive at princeton.edu:/pub/harnad/sg.comp.arch).

[Hausser, 1989] Hausser, R. (1989). *Computation of Language: An Essay on Syntax, Semantics and Pragmatics in Natural Man-Machine Communication.* Springer-Verlag, New York, NY.

[Hendrey, 1969] Hendrey, D. P. (1969). *Conditioned Reinforcement.* The Dorsey Press.

[Hexmoor, 1989] Hexmoor, H. (1989). An architecture for reactive sensor-based robot. In *NASA Goddard conference on AI*, Greenbelt, MD.

[Hexmoor, 1994] Hexmoor, H. (1994). What are Routines good for? In *AAAI Fall Symposium*, New Orleans, LA. Also available from SUNY at Buffalo, CS Department TR 94-07.

[Hexmoor, 1995a] Hexmoor, H. (1995a). Learning routines. In Wooldrige, M., Mueller, J., and Tambe, M., editors, *Agent theories, architectures and languages*. Springer Verlag. In Press.

[Hexmoor et al., 1993a] Hexmoor, H., Caicedo, G., Bidwell, F., and Shapiro, S. (1993a). Air battle simulation: An agent with conscious and unconscious layers. In *University of Buffalo Graduate Conference in Computer Science-93 (TR93-14)*. Dept. of Computer Science, SUNY at Buffalo, New York.

[Hexmoor and Kortenkamp, 1995] Hexmoor, H. and Kortenkamp, D. (1995). Issues on building software for hardware agents. In *Knowledge Engineering Review 10:3*. Cambridge University Press, London.

[Hexmoor et al., 1993b] Hexmoor, H., Lammens, J., Caicedo, G., and Shapiro, S. C. (1993b). Behavior Based AI, Cognitive Processes, and Emergent Behaviors in Autonomous Agents. In Rzevski, G., Pastor, J., and Adey, R., editors, *Applications of AI in Engineering VIII, Vol.2, Applications and Techniques*, pages 447–461. CMI/Elsevier.

[Hexmoor et al., 1992] Hexmoor, H., Lammens, J., and Shapiro, S. (1992). An autonomous agent architecture for integrating perception and acting with grounded, embodied symbolic reasoning. Technical Report CS-92-21, Dept. of Computer Science, SUNY at Buffalo, New York.

[Hexmoor et al., 1993c] Hexmoor, H., Lammens, J., and Shapiro, S. (1993c). An autonomous agent architecture for integrating "unconscious" and "conscious", reasoned behaviors. In *Computer Architectures for Machine Perception*, New Orleans, LA.

[Hexmoor et al., 1993d] Hexmoor, H., Lammens, J., and Shapiro, S. C. (1993d). Embodiment in GLAIR: A Grounded Layered Architecture with Integrated Reasoning. In *Florida AI Research Symposium*, pages 325–329. also available as SUNYAB CS Dept. TR93-10.

[Hexmoor and Nute, 1992] Hexmoor, H. and Nute, D. (1992). Methods for deciding *what to do next* and learning. Technical Report AI-1992-01, AI Programs, The University of Georgia, Athens, Georgia. Also available from SUNY at Buffalo, CS Department TR-92-23.

[Hexmoor et al., 1990] Hexmoor, H., Nute, D., and Underwood, B. (1990). PACE: A planning acting and control environment. In *3rd Annual Test Symposium*. Johns Hopkins University. Available by request from the first author (email:hexmoor@cs.buffalo.edu).

[Hexmoor and Shapiro, 1995] Hexmoor, H. and Shapiro, S. C. (1995). Building skillful agents. In Ford, K., Hoffman, R., and Feltivich, P., editors, *Human and Machine Expertise in Context*. In press.

[Hexmoor, 1995b] Hexmoor, H. H. (1995b). Smarts are in the architecture. In *AAAI Spring Symposium on Lessons Learned From Implemented Software Architectures for Physical Agents*, pages 116–122. AAAI press.

[Kaelbling, 1986] Kaelbling, L. (1986). An architecture for intelligent reactive systems. reasoning about actions and plans. In nd A. Lansky, M. G., editor, *Reasoning about Actions and Plans*. Morgan Kauffman.

[Kaelbling, 1988] Kaelbling, L. (1988). Goals as parallel program specifications. In *Proceedings of AAAI-88*. Morgan Kaufman.

[Kaelbling, 1990] Kaelbling, L. (1990). *Learning in Embedded Systems*. PhD thesis, Stanford University. Also as Teleos TR 90-04, Menlo Park, CA.

[Kaelbling and Rosenschein, 1990] Kaelbling, L. and Rosenschein, S. (1990). Action and planning in embedded agents. In Maes, P., editor, *Designing Autonomous Agents*, pages 35–48. MIT Press.

[Kumar, 1994] Kumar, . D. (1994). *From Beliefs and Goals to Intentions and Actions: An Amalgamated Model of Inference and Acting*. PhD thesis, Technical Report 94-04, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY.

[Kumar, 1989] Kumar, D. (1989). An integrated model of acting and inference. In Kumar, D., editor, *Current Trends in SNePS–Semantic Network Processing System: Proceedings of the First Annual SNePS Workshop*, pages 55–65, Buffalo, NY. Springer-Verlag.

[Kumar et al., 1988] Kumar, D., Ali, S., and Shapiro, S. C. (1988). Discussing, using and recognizing plans in SNePS preliminary report - SNACTor: An acting system. In *Proceedings of the Seventh Biennial Convention of South East Asia Regional Confederation*, pages 177–182, New Delhi, India. Tata McGraw-Hill.

[Kumar and Shapiro, 1991] Kumar, D. and Shapiro, S. C. (1991). Modeling a rational cognitive agent in SNePS. In Barahona, B., Pereira, L. M., and Porto, A., editors, *EPIA 91: 5th Portugese Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence 541*, pages 120–134. Springer-Verlag, Heidelberg.

[Laird et al., 1991] Laird, J., Huka, M., Yager, E., and Tucker, C. (1991). Robo-soar: An integration of external interaction, planning, and learning, using soar. In *Robotics and Autonomous Systems*.

[Laird et al., 1987] Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64.

[Lakoff, 1987] Lakoff, G. (1987). *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind.* University of Chicago Press, Chicago, IL.

[Lammens et al., 1995] Lammens, J., Hexmoor, H., and Shapiro, S. (1995). Of elephants and men. In Steels, L., editor, *NATO ASI Series on biology and technology of intelligent autonomous agents, Volume 144*. Springer-Verlag.

[Langley et al., 1994] Langley, P., Iba, W., and Shrager, J. (1994). Reactive and automatic behavior in plan execution. In *Second International Conference on AI Planning Systems.* AAAI Press.

[Langley et al., 1991] Langley, P., McKusick, K., and Allen, J. (1991). A design for the ICARUS architecture. In *ACM SIGART Bulletin*, pages 104–109. ACM publications.

[Langton, 1989] Langton, C., editor (1989). *Artificial Life.* Addison Wesley.

[Larid et al., 1986] Larid, J., Rosenbloom, P., and Newell, A. (1986). Chunking in soar: The anatomy of a general learning mechanism. In *Machine Learning 1*, pages 11–46. Kluwer Academic Publishers.

[Levy, 1992] Levy, S. (1992). *Artificial Life.* Vintage Books.

[Lin, 1991] Lin, L.-J. (1991). Programming robots using reinforcement learning and teaching. In *Proceedings of AAAI-91*, pages 781–786.

[Lin, 1992] Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning, and teaching. In *Machine Learning 8*, pages 293–321.

[Maes, 1990] Maes, P. (1990). Situated agents can have goals. In Maes, P., editor, *Designing Autonomous Agents*, chapter 4, pages 49–70. MIT Press.

[Maes, 1991] Maes, P. (1991). Action selection. In *Proceedings of Cognitive Science Society Conference*.

[Maes and Brooks, 1990] Maes, P. and Brooks, R. (1990). Learning to coordinate behaviors. In *AAAI-90*, pages 796–802.

[Mahadaven and Connel, 1991] Mahadaven, S. and Connel, J. (1991). Scaling reinforcement learning to robotics by exploiting the subsumption architecture. In *Maching Learning Workshop*, pages 328–332.

[Mani and Uhr, 1992] Mani, G. and Uhr, L. (1992). Acquiring rules for need-based actions aided by perception and language. In *Proceedings of the 14th Annual Cognitive Science Conference*.

[Mataric, 1993] Mataric, M. J. (1993). Designing emergent behaviors: From local interactions to collective intelligence. In *From Animals to Animats 2*, pages 432–441.

[McClelland et al., 1986] McClelland, J. L., Rumelhart, D. E., and Hinton, G. E. (1986). The appeal of parallel distributed processing. In Rumelhart, D., McClelland, J., and the PDP Research Group, editors, *Parallel Distributed Processing*, chapter 1, pages 3–44. MIT Press, Cambridge MA.

[McDermott, 1991] McDermott, D. (1991). Robot planning. Technical Report CS-861, Yale University.

[Miller, 1956] Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. In *The Psychological Review*, number 63-2, pages 81–97. The American Psychological Association, Inc., Washington DC.

[Minton, 1988] Minton, S. (1988). Effective search control strategy: An explanation-based approach. Technical Report CMU-CS-88-133, Carnegie Mellon University.

[Mitchell et al., 1991] Mitchell, T., Allen, J., Chalasani, P., Cheng, J., Etzioni, O., Ringuette, M., and Schlimmer, J. (1991). Theo: A framework for self-improving systems. In VanLehn, K., editor, *Architectures for Intelligence*, pages 323–356. Lawrence Erlbaum.

[Mitchell, 1990] Mitchell, T. M. (1990). Becoming increasingly reactive. In *Proceedings of AAAI-90*, pages 1051–1058. AAAI.

[Mitchell et al., 1986] Mitchell, T. M., Keller, R., and Kedar-Cabelli, S. (1986). Explanation based generalization: A unifying view. In *Machine Learning*, volume 1, pages 47–80. Kluwer Acedemic Publishers, Boston.

[Nehmzow, 1992] Nehmzow, U. (1992). *Experiments in Comptence Acquisition for Autonomous Mobile Robots*. PhD thesis, University of Edinburgh.

[Newell, 1981] Newell, A. (1981). The knowledge level. *Artificial Intelligence*, 2(2):1–20.

[Nilsson, 1992] Nilsson, N. J. (1992). Toward agent programs with circuit semantics. Technical Report STAN-CS-92-1412, Stanford University, Stanford, CA.

[Norman, 1981] Norman, D. (1981). Categorization of action slips. In *Psychological Review*, volume Volume 88 Number 1, pages 1–15. The Psychological Association, Inc, Washington DC.

[Payton, 1986] Payton, D. (1986). An architecture for reflexive autonomous vehicle control. In *Proceedings of Robotics Automation*, pages 1838–1845. IEEE.

[Pollock, 1989] Pollock, J. (1989). *How to Build a Person*. MIT Press.

[Pollock, 1992] Pollock, J. (1992). New foundations for practical reasoning. In *Minds and Machines*.

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

[Rosenschein and Kaelbling, 1986] Rosenschein, S. and Kaelbling, L. P. (1986). The synthesis of digital machines with provable epistemic properties. In *Proceedings of Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 83–89.

[Russell, 1991] Russell, S. (1991). An architecture for bounded rationality. In *ACM SIGART Bulletin*, pages 146–150. ACM publications.

[Schmidt, 1975] Schmidt, R. A. (1975). A schema thoery of discrete motor skill learning. In *Psychological Review*, volume 82-1, pages 225–259. The American Psychological Association, Inc, Washington DC.

[Scholnick and Friedman, 1987] Scholnick, E. and Friedman, S. (1987). The planning construct in psychological literature. In *BluePrints for Thinking*, pages 3–38. Cambridge University Press, London.

[Schoppers, 1987] Schoppers, M. J. (1987). Universal plans for unpredictable environments. In *Proceedings 10th IJCAI*, pages 1039–1046.

[Schoppers, 1989] Schoppers, M. J. (1989). *Representation and Automatic Synthesis of Reaction Plans*. PhD thesis, University of Illinois at Urbana-champaign.

[Segre and DeJong, 1985] Segre, A. and DeJong, G. (1985). Explanation-based manipulator learning: Acquisition of planning ability through observation. In *IEEE Robotics and Automation*, pages 555–560.

[Shapiro, 1990] Shapiro, S. C. (1990). Cables, paths, and 'subconscious' reasoning in propositional semantic networks. In *Principles of Semantic Networks*. Morgan Kaufman.

[Shapiro and Rapaport, 1986] Shapiro, S. C. and Rapaport, W. J. (1986). SNePS considered as a fully intensional propositional semantic network. In *Proc. Fifth National Conference on Artificial Intelligence*, pages 278–283, Los Altos, CA. Morgan Kaufmann. Superseded by [Shapiro and Rapaport, 1987].

[Shapiro and Rapaport, 1987] Shapiro, S. C. and Rapaport, W. J. (1987). SNePS considered as a fully intensional propositional semantic network. In Cercone, N. and McCalla, G., editors, *The Knowledge Frontier*, pages 263–315. Springer–Verlag, New York.

[Shen, 1989] Shen, W.-M. (1989). *Learning from the Environment Based on Actions and Percepts*. PhD thesis, Carnegie Mellon University.

[Shepard and Metzler, 1971] Shepard, R. and Metzler, J. (1971). Mental rotation of three-dimensional objects. In *Science*. Consulting Psychlogists Press.

[Shiffrin and Dumais, 1981] Shiffrin, R. M. and Dumais, S. (1981). The development of automatism. In *Cognitive Skills and Their Acquisition*, pages 111–140. Earlbaum Associates, Hillddale, NJ.

[Shiffrin and Schneider, 1977] Shiffrin, R. M. and Schneider, W. (1977). Controlled and Automatic Human Information Processing: II. Perceptual Learning, Automatic Attending, and a General Theory. In *Psychological Review*, volume 84-2, pages 127–190. The American Psychological Association, Inc, Washington DC.

[Shrager, 1990] Shrager, J. (1990). Commonsesne perception and the psychology of theory formation. In *Computational Models of Sceintific Discovery and Theory Formation*, pages 437–470. LawreenceEarlbaum Associates, Hillsdale, NJ.

[Shrager, 1994] Shrager, J. (1994). Personal communication.

[Siegler and Crowley, 1994] Siegler, R. and Crowley, K. (1994). Constraints on learning in nonpriviledged domains. In *Cognitive Psychology*, volume 27, pages 194–226. Academic Press, Inc.

[Siegler and Jenkins, 1989] Siegler, R. and Jenkins, E. (1989). *How Children Discover New Strategies*. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

[Simmons, 1990] Simmons, R. (1990). An architecture for coordinating planning, sensing, and action. In *Proceedings the DARPA workshop*, pages 292–297.

[Smith and Stevens, 1994] Smith, T. and Stevens, G. (1994). Emergence, self-organization, and social interaction: Arousal- dependent structure in social systems.

[Suchman, 1988] Suchman, L. A. (1988). *Plans and Situated Actions: The Problem of Human Machine Communication*. Cambridge University Press.

[Sutton, 1984] Sutton, R. (1984). *Temporal Credit Assignment in Reinforcement Learning.* PhD thesis, University of Massachusetts.

[Sutton, 1988] Sutton, R. (1988). Learning to predict by the methods of temporal differences. In *Machine Learning 3*, pages 3–44. Kluwer Academic publishers, Boston.

[Watkins, 1989] Watkins, C. (1989). *Learning from Delayed Rewards.* PhD thesis, King's College, Cambridge, UK.

[Whitehead, 1992] Whitehead, S. (1992). *Reinforcement Learning for the Adaptive Control of Perception and Action.* PhD thesis, University of Rochester. Technical Report 406.

[Wilkins, 1984] Wilkins, D. E. (1984). Domain-independent planning: Representation and plan generation. *Artificial Intelligence*, 22:269–301.

[Winston, 1975] Winston, P. H. (1985 (orig. 1975)). Learning structural descriptions from examples. In Brachman, R. J. and Levesque, H. J., editors, *Readings in Knowledge Representation*, pages 141–168. Morgan Kaufmann, San Mateo CA.