Expertise in Context

Human and Machine

Edited by

Paul J. Feltovich
Kenneth M. Ford
&
Robert R. Hoffman

AAAI Press / The MIT Press
Menlo Park, California / Cambridge, Massachusetts / London, England

CHAPTER 16

Integrating Skill and Knowledge in Expert Agents

Henry Hexmoor & Stuart C. Shapiro

Introduction

An empirical characteristic of the development of expertise is the transformation of deliberative, "conscious" activity to more automatic forms. In this chapter, we will discuss knowledge organization and representation for skills, and the integration of skills with knowledge. We will illustrate our concepts by discussing a few implemented agents in hardware and software. An underlying theme for our formal representations is *embodiment* of knowledge. "We define *embodiment* as the notion that the *representation and extension of concepts* is in part determined by the *physiology* (the bodily functions) of an agent, and in part by the *interaction* of the agent with its environment" (Hexmoor, Lammens and Shapiro 1993, p. 326). With this prior definition, all computer representations are embodiments of their hardware and how they are used in interaction with other agents. We intend our definition to apply to agents that physically manipulate the environment and themselves have physically moving parts. Furthermore, we use our definition as a prescription for developing representations instead of a description of all possible representations. We have developed an autonomous agent architecture that incorporates action and perception which are embodied in terms of the moving parts and the dynamics of the moving parts when the agent interacts with the world.

It seems clear that book knowledge differs from practical knowledge. For example, an automobile driver uses a combination of his book knowledge of driving rules and tips as well as practical knowledge in his actual driving. In this chapter, we will not focus on this distinction. Instead, we are concerned with the balance of using knowledge and skill over time as two components of expertise. Henceforth, skills in this chapter refer to motor skills. For us,

skills are that part of expertise that is acquired through practice. Furthermore, once a skill is acquired it is largely inaccessible to explanation, without reflection and reasoning.

This inaccessibility leaves an unclear breakdown of expertise into book knowledge and practical knowledge. By focusing on the distinction between knowledge and skills, one can see that the expertise of a medical specialist lies mostly in his knowledge about his area of specialization, while the expertise of a surgeon includes his skills in surgical procedures. The expertise of an athlete is mostly his skills. Skills are often hard to put into words. Athletic coaches often speak in strange jargon that does not refer to body parts. Examples from bowling are over-turn, "applying too much spin to the ball and not enough finger lift"; loafing, "not lifting or turning the ball properly with the result that the ball lags and lacks action", and yanking the shot, "hanging onto the ball too long and pulling it to across the body" (Taylor 1991). But what are skills and how does an agent possess them? Why is it sometimes hard to verbalize skills? Can skill knowledge be represented by a production system or a declarative representation? Anderson's (1983) Act* assumes the answer is yes to the last question. Instead of empirical tests and psychological investigation, we build computer models that seem to mimic the internal mechanisms of an expert in acquiring skill and knowledge (i.e., expertise). We hope that our computational models and synthetic agents will provide an alternative mechanism for exploring issues about knowledge and skill which is somewhat independent of ethnomethodology and biological organisms.

Traditionally, expert systems addressed the knowledge component of expertise. These systems were often designed with an impoverished ontology of percepts and actions (e.g., IF symptoms *A* and *B* are present THEN prescribe medication *C*). We posit that the knowledge used in expertise is necessarily embodied in the expert and ought to be represented in terms of the agent's physiology and the interaction of the agent with its environment. We don't use embodiment as merely meaning that the agent has a body. Knowledge resides in various parts of the expert agent (e.g., cerebellum versus cerebral cortex), and is used in different cognitive processes.

Once a skill is acquired, it changes with experience. However, natural language plays an important role for skills that are primarily acquired through communication. We hypothesize that the human brain's natural language comprehension mechanism translates representations to embodied representations which are more natural for representing skills. We believe that many current implemented natural language comprehension systems ignore the role of body-centered understanding. With this hypothesis we believe we need to model multi-level representations within an agent. This chapter focuses on embodied representations of knowledge that are "natural" for the agent, and leaves the natural language abstractions which make communication easier as a separate problem.

We believe that in order to understand expertise, it is important to understand the ability to use, coordinate, and learn the knowledge involved in different cognitive processes that exist in different parts of the expert agent's body. In our investigations, we are examining the mechanisms for acquiring knowledge and skill, the migration of knowledge in the agent, and the nature of expertise, by designing and implementing an agent architecture. In this chapter, we describe our architecture and give an ontology of concepts in our approach. We also describe a representational formalism for encoding skills. We then describe "agents" that have been developed using our architecture. We demonstrate several of our concepts about expertise concretely using our implemented agents. This will be followed by a discussion of knowledge migration in our architecture and implemented agents as examples. We will conclude this chapter by pointing out how agents use and gain knowledge and skills with our architecture.

Architecture

To model an expert, we have developed an architecture called grounded layered architecture with integrated reasoning, GLAIR (Hexmoor, Lammens and Shapiro 1992), schematically presented in figure 1.

The figure shows three distinct levels: knowledge, perceptuo-motor, and sensory-actuator. These levels provide a framework for modeling distinct types of behavior generation. The Knowledge Level (KL) is considered to contain the agent's "conscious" beliefs and plans. The KL is the only level in GLAIR accessible to natural language use and generation. This accessibility to natural language sets the KL apart from the other levels. We use the SNePS Knowledge Representation and Reasoning system (Shapiro and Rapaport 1992; Shapiro and Group 1992) to represent the knowledge at the KL. Knowledge representation and reasoning systems such as SNePS are general purpose tools to explicitly and symbolically encode the contents and processes of a cognitive agent's "mind." These systems have formal properties that help in the analysis of what is represented.

The other two levels, Perceptuo-Motor Level (PML) and Senson-Actuator Level (SAL), are considered to contain "unconscious" processes. The PML is used to model skills. The SAL models the actuator hardware and all unchanging and least complex agent-centered processing of actuator input and output. For detailed discussions of GLAIR and comparisons with competing architectures see (Hexmoor *et al.* 1992).

We have adopted an ontology of terms used in various parts of GLAIR which primarily reflects our choice of levels for modeling various parts of an autonomous agent. In the next section, we describe our terms.

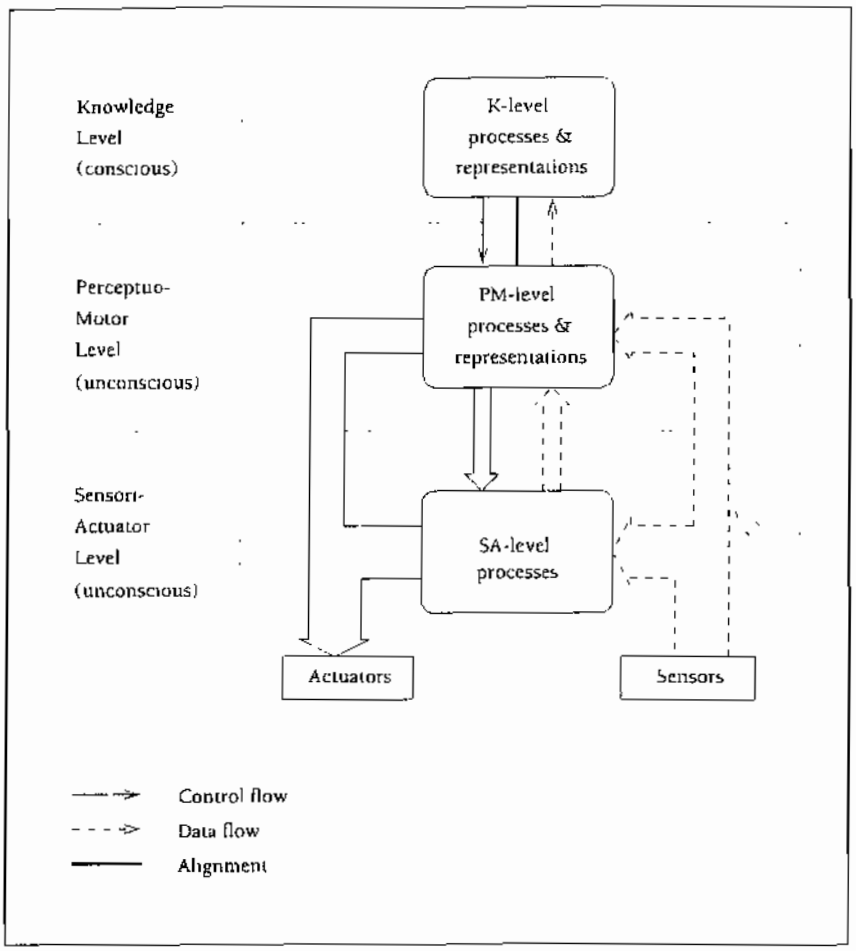

Figure 1. Schematic representation of the GLAIR architecture. Width of control and data paths suggests the amount of information passing through (bandwidth). Sensors include both world-sensors and propio-sensors.

An Ontology

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 in this chapter the reader can think of them as having their common sense meanings. The entities modeled at the PML are *sensations, actuators, commands, behaviors,* and *tendencies.* We use *sensations* as atomic terms used in describing perception. *Actuators* are the effecting parts of the agent (e.g., muscles or motors). We assume that actuators can be commanded independently.[1] 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. By a *behavior* we mean a cluster of possibly simultaneous commands, at most one for each 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.

In order to transfer information between levels, either all terms in the levels have a one-to-one correspondence, or some terms do not correspond to anything in the other levels. For example, a sequence of commands at the PML may map into a single primitive act (see Kumar and Shapiro 1991) at the KL. For another example, intentions do not map to anything at the PML because intentions at the KL refer to a decision to execute that action. At the PML there is no need to consider decisions about actions.

When there is a one-to-one correspondence, there is a correspondence between terms in the Knowledge Representation and Reasoning (KRR) system on one hand and perceived objects, properties, events, and motor capabilities on the other hand. We call this correspondence *alignment*.

Obviously, beyond what the ontology dictates, there is no domain-independent way of deciding what goes into each level. We consider the locus of knowledge and skills to be dynamically changing between levels. *Perceptual reduction* and *action elaboration* are two salient features that guide location of knowledge and skills. Perceptual reduction is the grouping of perceptual data into perceptual concepts at higher levels. Action elaboration is the expansion of motor actions into finer actuator controls at lower levels.

In addition to the changing locus of knowledge and skill, the locus of control also changes. We will discuss exact mappings between levels in the context of projects and domains later in this chapter.

In the following subsection we describe Perceptuo-Motor Automata (PMA) as our representational tool in the PML. PMA can be considered to be a specialized production system as much as Soar (Laird 1987) is a specialized production system. Our tool is specialized for modeling situated actions in intelligent agency. Many features, like concurrency of behavior activation, distinguish it from similar systems.

Perceptuo-Motor Automata

In order to generate commands for actuators, we have developed a representation mechanism and a modeling tool for representing and generating behaviors (metaphorically at the "unconscious" level) of an autonomous agent (Hexmoor 1992). PMA is a tool used in modeling the PML behavioral components of a GLAIR-agent. Behavioral modules that PMA (as a tool) models are also called PMA. Therefore, we might refer to a PMA for *chewing* behavior and a PMA for *walking*. Our use of the term will be clear from the context.

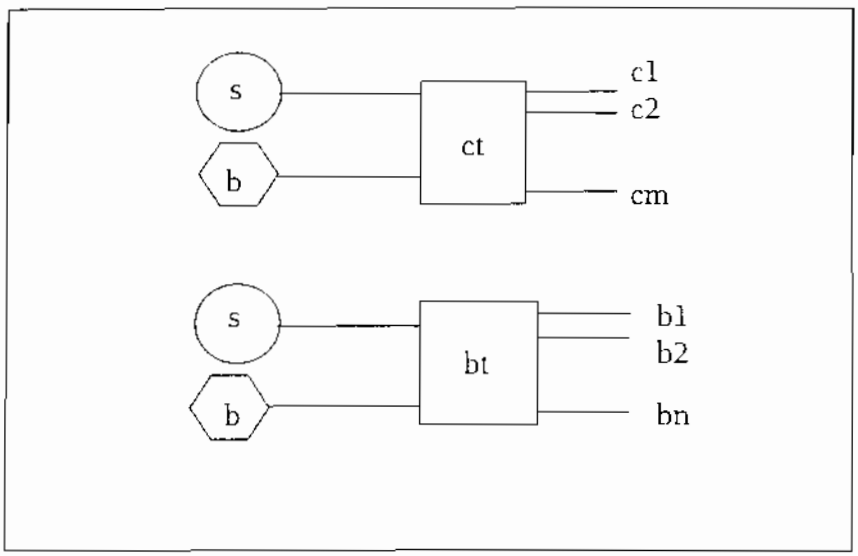

Figure 2 A command transition is shown on the left that maps a Situation, s1, and Behavior, b1, to a Command, c1. A Behavior transition is shown in the right that maps a pair of Situations, s2, and Behavior, b2, to a Behavior, b3

PMA as a system is designed to constantly produce signals/impulses for the actuators. We consider these signals/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 would be a fraction of a second in duration. In the context of a GLAIR-agent, PMA models rapid and automatic behaviors of the agent.

We assume that agent's have a finite number of actuators. An actuator is an effecting part of the agent that can be independently commanded. A behavior is a set of commands, one for each 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 SAL.

The set of behavioral modules that PMA models in a domain is formally represented by <Sensations, Commands, Command Transition, Behaviors, Behavior Transition, Rewards>. For example, a Sensation for a mobile robot that walks in building hallways can be its *distance to the wall*. In Air Battle Simulation (ABS) (Hexmoor *et al.* 1993), *relative distances from the enemy in X, Y, and Z axes* are Sensations. We assume that all possible situations for a PMA can be specified by a combination of one or more Sensations. We call a pattern of Sensations that is used for input to a PMA a Situation. In ABS, instantiations of <*distanceX, distanceY, distanceZ, orientation*> are Situations (e.g.,

<close-front, close-right, close-above, parallel> is a Situation). In summary, PMA input is a Situation, an *n*-tuple of Sensations. Commands are a set of primitive activations that can be sent to actuators. Command Transitions (CT) are transformation functions that produce commands for execution for inputs of situation or situation/behavior pairs. This is shown in figure 2. CTs are functions since the output of a CT is a set of one or more Commands. The simplest CTs are mappings of Sensations to Commands (Sensations $\mapsto$ Command). 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.

CTs use Behaviors and previous Commands in producing output. When the latest sensations along with the current Behavior match a command transition, that command transition activates a command which is then executed. Our implementation of PMA is object-oriented and written in Common Lisp and C. Our descriptions of PMA operation therefore consider CTs as objects that can process their input and activate output nodes. CTs may take into account an estimated accumulated reward for Commands. We call the latter "tendencies." Tendencies are computed using reinforcement based learning techniques (Sutton 1988). Below is a list of Command Transition types.

- Sensations $\mapsto$ Command
- Previous command X Sensations $\mapsto$ Command
- Behavior X Sensations $\mapsto$ Command
- Tendency X Sensations $\mapsto$ Command
- Previous-command X Behavior X Sensations $\mapsto$ Command
- Tendency X Behavior X Sensations $\mapsto$ Command
- Previous-command X Behavior X Tendency X Sensations $\mapsto$ Command

Behaviors partition a PMA into smaller PMAs, each a set of command transitions. Behavior Transitions (BT) are transformations that update the Behavior in effect. 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 PML without a "conscious" decision. BT is Behavior1 X Sensations $\mapsto$ Behavior2 where Behavior1 and Behavior2 are Behaviors.

Rewards are used for learning. Rewards are static "goodness values" associated with situations. Estimated accumulated rewards are estimations of accumulations of rewards over time used in delayed credit assignment. These are determined prior to PMA execution and remain unchanged throughout.

Behaviors, Behavior Transitions, and Rewards in a PMA can be empty. The simplest class of PMA is that in which all three of these elements are empty. Below is the pattern 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). The following is the pattern for all possibilities for CT types.

[Previous command X] [Behavior X] [Tendency X] Sensations ↦ command set

The capacity to produce asynchronous and concurrent behaviors and concurrent Commands for actuators allows the modeling of certain interactions with the agents in the world. What is interesting is: (a) the serendipity of behavior combinations unbeknownst to the agent (i.e., emergent behaviors), and (b) the way the agent makes use of the successful interactions in the world by various kinds of learning processes. PMA is used as the agent's lowest level mechanism to learn from the positive interactions in the world.

In the following section, we describe examples of agents we are developing that illustrate the principles of the GLAIR architecture. Each agent is designed to exemplify different features of our architecture.

GLAIR Agents

We have developed several agents among which are a player of a video-game Air Battle Simulation (ABS) named Gabby, for GLAIR Air Battler, and an autonomous mobile robot named Gerry (the "G" stands for GLAIR and the two "r"s stand for "roving robot"). Gerry and Gabby differ in their mapping of terms between the KL and the PML. This difference reflects their different modes of interaction with the world. We conjecture that Gabby is more "mindful" of its actions and monitors its actions. We need to further analyze the mapping of terms between the KL and the PML to understand the modes of interactions.

All agents display a variety of integrated behaviors. We distinguish between deliberative, reactive, and reflexive behaviors. These behaviors are mostly exhibited by the KL, the PML, and the SAL respectively. As we move down from the KL to the PML and the SAL, computational and representational power is traded for better response time and simpler control. The agent learns from and automates its interactions with the environment.

Gabby and Air Battle Simulation

Figure 3 schematically presents the structure of the GLAIR agent that plays the ABS video-game. We will refer to this agent as Gabby. Before it starts learning, Gabby does not have a PMA and therefore uses "conscious" level reasoning (i.e., SNePS rules; Kumar and Shapiro 1991), to decide what move to make. Once transitions are learned and cached in a PMA, Gabby uses the PMA for deciding its next move whenever possible. By adding learning strategies, a PMA is developed that caches moves decided at the KL for future

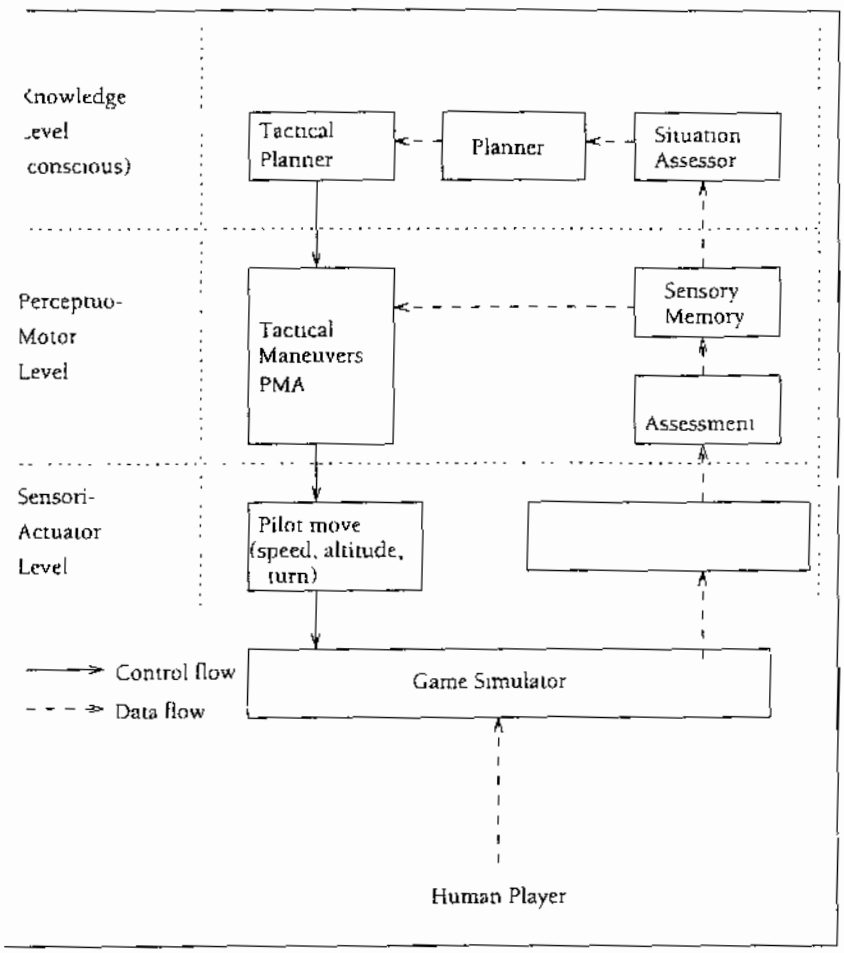

Figure 3 Schematic representation of the Air Battle Simulation playing GLAIR agent, Gabby.

use. Here again, learning is used to mark PMA moves that proved unwise and to reinforce moves that turn out to be successful. Gabby demonstrates real time behaviors and the interlevel alignment mechanism.

The ABS video game runs on SparcStations and simulates World War I-style air plane dog-fights between Gabby and a human player. The game display consists of a main window giving a horizontal view, a side window giving a vertical view, and two "damage reports." This is shown in figure 4. The control panel window is shown in figure 5. The human 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 are displayed by

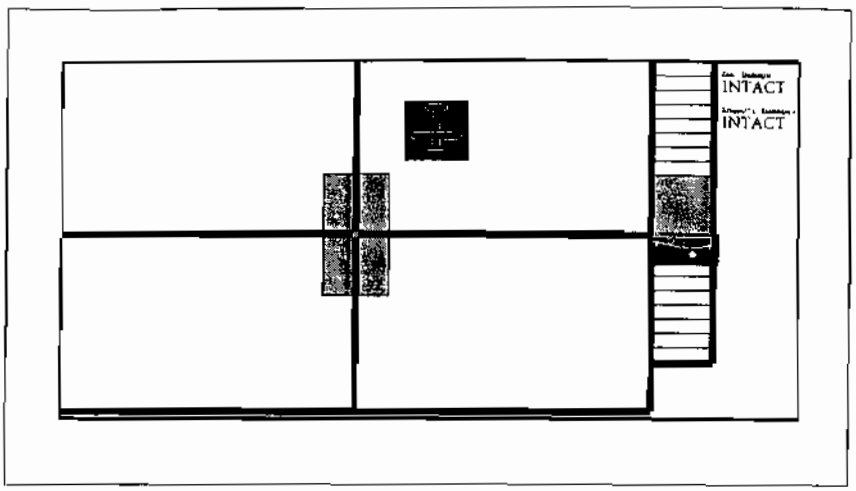

Figure 4. Air Battle Simulation game windows. Gabby's plane is indicated by the drawing of a plane in the upper right quadrant of the main window and by the drawing of a plane in the side window. This figure shows Gabby fleeing, flying parallel to and at a higher altitude than the human. The shaded regions denote shooting range. If Gabby's plane appears in both shaded regions, whichever plane is facing the other (possibly both) will fire.

the drawing of a plane in the main window, and its height relative to the human's plane is indicated by the drawing of a plane in the side window. The condition of the human'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 human makes a separate decision about when to fire.

The human player uses the control panel to choose a move, which comprises a combination of changing altitude, speed, and direction. When the human 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 players' planes are destroyed.

Gerry

Gerry began as an Omnibot 2000 toy robot (Amherst Systems, Inc.). Gerry is a 2.5 feet tall mobile robot with two independently driven wheels, an in-

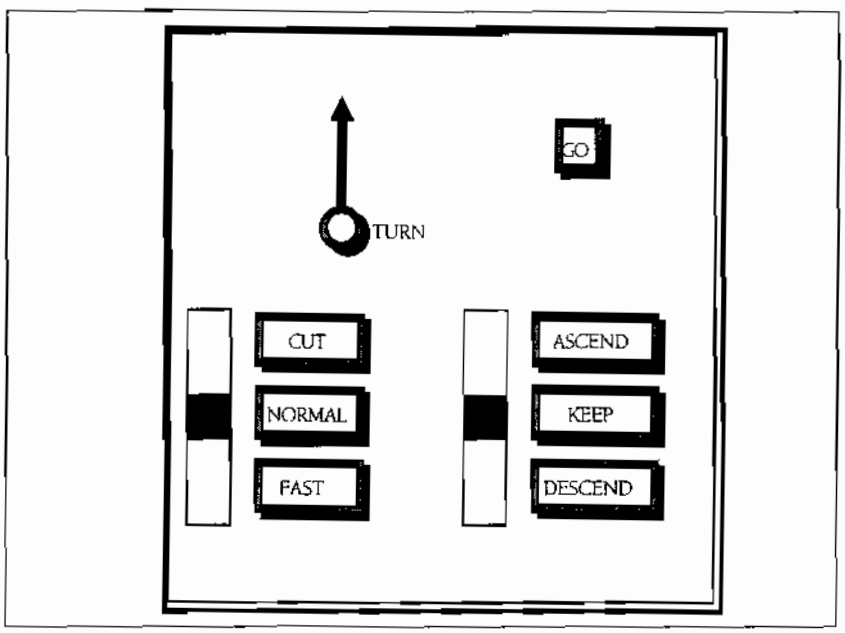

Figure 5. Air Battle Instrument Panel window. To select/change a move, the human player pushes one of the buttons in each column and adjusts the TURN dial by repeatedly clicking on it. CUT, NORMAL, and FAST are used for speed selection.

frared transmitter, four infrared sensors, a front and rear bumper, and four corner whiskers controlled by a 6.270 board developed at MIT for use in their 6.270 class. These boards have a 6811 microprocessor with multitasking capabilities. 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 the PML of Gerry, behaviors are implemented using PMA.

Currently, Gerry has PMAs and some basic knowledge at the KL for searching for another robot, Garry. Garry is one foot tall, has two independently driven wheels, an infrared transmitter, two infrared sensors, and three front bumper/whiskers controlled by a 6.270 board. When a whisker or a bumper senses contact with an external object, Garry performs a reflex. Garry is a Lego robot.

Figure 6 shows that Gerry's KL knowledge consists of three separate components 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 is implemented by having concepts at the KL correspond to patterns of stimulus at the PML. For example, "no-garry" at

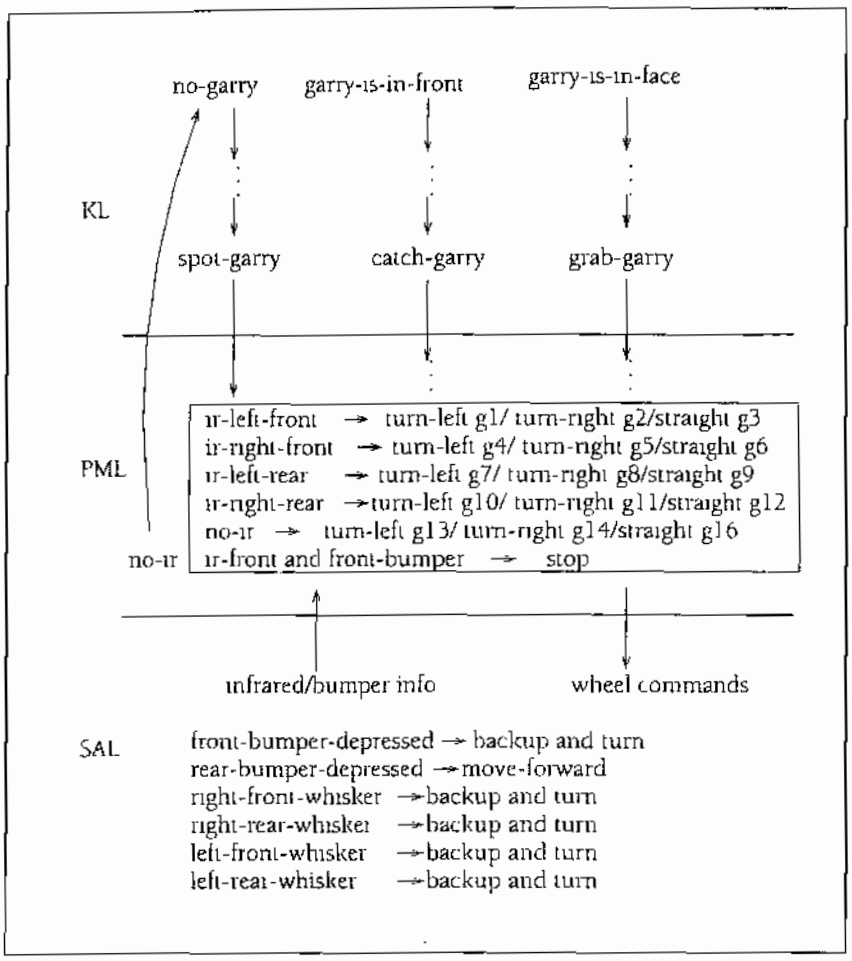

Figure 6 Schematic representation of the Gerry's GLAIR components. The box in the PML shows a PMA for spotting Gerry. There are others for catching up with Garry and grabbing Garry Variable gi next to each action in the PMA box represents goodness of action in the context of a corresponding situation.

the KL corresponds to the pattern of no infrared being sensed at the PML. At the KL, no concepts for specific infrared signals as in percetuo-motor level exist. Each action in the knowledge level is expanded into a PMA in the perceptuo-motor level. This is an implementation of action elaboration in which actions at the PML are finer compared to the KL actions. As shown in Figure 6, action elaboration is implemented by mapping an action in the KL to a PMA. For example, "spot-garry" is shown to map to a PMA in the PML,

h is shown in the solid box Transitions in the PMA map infrared situa-
to commands With Gerry, each situation in the PMA is mapped to all
nands What determines applicability of a command, in a given situa-
is its goodness level shown by gi in figure 6.

the SAL, Gerry has six reflexes for avoiding obstacles Figure 6 shows
monic forms of these reflexes The actual implementation of actuator
mands is given in terms of wheel commands.

oncurrent behaviors at the PML are allowed, as are concurrent reflexes.
iviors and reflexes produce wheel commands All wheel commands are
n a priority level when they are generated. The PML wheel commands
: lower priority than the ones for reflexes One exception is that behav-
have higher priority than reflexes if a reflex needs to be suppressed For
nple, to catch Garry, Gerry needs to have its front bumper touch Garry
sense high levels of infrared radiation with its front sensors. Normally,
:hing an object with the front bumper triggers a reflex to backup and
1 However, we want to suppress this reflex. There is a module at the SAL
arbitration and combination of all wheel commands This module exam-
; flags for suppression and, if reflexes are not suppressed, it sums the ef-
of reflex-generated wheel commands and passes it to the wheels. If no
ex is active, it sums all behavior-generated wheel commands and passes
result to the wheels If reflex suppression is turned on, it ignores reflexes
l attends to the behavior generated commands.

We have programmed PMAs and the knowledge at the KL so Gerry will
k for Garry and will stop when it finds Garry At the PML, Gerry is to re-
ve nondeterminacy using reinforcement based learning Gerrys actions
1 be controlled either completely by the KL or by the PML.

In the following section we describe learning in a GLAIR level that is based
the contents and influence of a neighboring level. We call this type of
rning knowledge migration. We will use Gerry and Gabby to illustrate this
.rning strategy

Knowledge Migration

1 important feature of GLAIR is that it allows knowledge to migrate from
ie level to another and in doing so change the representation in order to be
insistent with the representation used at the new level The underlying as-
umption here is that parts of an agent's knowledge about its world have
iatural" loci either as deliberative knowledge, as reactions, or as reflexes As
ie agent interacts with the world, the knowledge may be gained at some
vel and later find its "natural" locus at a different level and migrate there In
ie following subsection, we will describe learning in the PML from the KL.

"Conscious" to "Unconscious" Migration

"What to do next" is a the ability to choose an action to perform, given the current situation (Hexmoor and Nute 1992). Our migration of knowledge from the "conscious" to "unconscious" is limited to "what to do next." As in *automaticity,* migration of knowledge about actions from the KL to the PML makes the agent's reactions in the environment faster and less deliberate.

> In novel situations or in situations requiring moment-to-moment decisions, controlled processing may be adopted and used to perform accurately, though slowly Then as the situations become familiar, always requiring the same sequence of processing operations, automatic processing will develop, attention demands will be eased, other controlled operations can be carried out in parallel with the automatic processing, and performance will improve (Shiffrin 1977, p 161).

Knowledge compilation (Anderson 1983) is a twofold procedure of *proceduralization* to convert declarative knowledge into productions and *composition* to combine several productions into one. Our knowledge migration is similar to Anderson's knowledge compilation in that the migration process produces PMA transitions which are a kind of production rules However, unlike Anderson's knowledge compilation, our transitions in the PMA are maintained separately from the initial knowledge.

We describe two implementations of knowledge migration In both, migration transforms representations of knowledge into PMA transitions

In the process of knowledge migration, Gabby drops all the intermediate reasoning and only retains the stimuli/response pair in the PML. Furthermore, Gabby adds what it learns in each novel situation to the existing structure of the PML By enriching existing structures in PML and extracting/reformatting knowledge existing at the KL we feel we have modeled a form of *chunking* (Miller 1956).

At the PML, commands have a one to one correspondence with KL actions, and behaviors have a one-to-one correspondence with KL goals. Consider C and B in the PML to be unique counterparts of A and G respectively in the KL and S to be the current situation If $S \times B \mapsto C$ is not already a CT in the PML, it is made one. B is the behavior in effect and remains so until a new goal is selected at the "conscious" level. If a new G (corresponding to new behavior B' in the PML) is transmitted to the PML, the behavior in effect is updated to B'. If $S \times B \mapsto B'$ is not a BT, it is made one.

Example of Migration in the Agents. We started Gabby with an empty PMA. Whenever an appropriate PMA CT existed, it was used. Otherwise, the KL handled command generation, and a new PMA was created. As the game was played, we observed that the agent became more reactive since the PMA was used more often to generate behaviors instead of the KL.

Figure 7 shows a small, typical sample of PMA transitions Gabby learned while playing ABS. Many other transitions were learned Only four CTs and

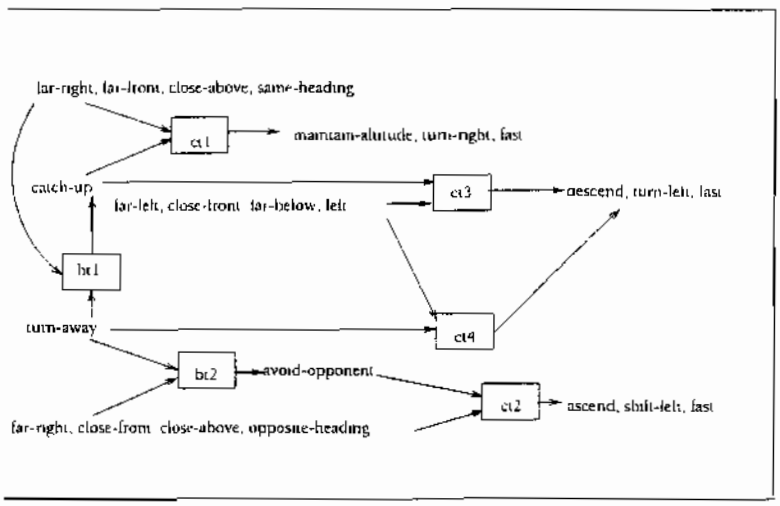

Figure 7. Sample of Learned Command and Behavior Transitions.

Ts are shown in the figure. We narrowly consider knowledge of "what next" (Hexmoor and Nute 1992; i.e., given the current situation what should be performed) to illustrate automaticity In GLAIR, migration owledge about actions from the knowledge level to the PM-level can be ht of as a form of automaticity.

en Gabby in the knowledge level determines an action, it submits a f an action A and a goal G to the PML At the PM-level, commands are one associated with K-level actions, and behaviors are one to one as- ed with K-level goals. Let's consider command C and behavior B to be nque counterparts of action A and goal G respectively, and situation S the current situation If $S \times B \mapsto C$ is not a CT, it is made one B is the ior in effect and unless a new goal is selected at the conscious level, it is to be the behavior in effect If a new G (corresponding to new be- B' in the PM level) is transmitted to the PM-level, the behavior in is updated to be B' If $S \times B \mapsto B'$ is not a BT, it is made one

igure 7, if the behavior is turn-away and the situation of the opponent

execute. *ct3* and *ct4* are two other command transitions that are learned and the transitions are self-evident from the figure.

Instead of learning through repetition as it occurs with automaticity, mappings for every novel situation in PMA were stored in the PML (i.e., one trial learning). After the first game ended, Gabby began succeeding games with partially learned PMAs and continued learning. The game involved highly recurring patterns of interaction. Interactions in situations perceived as near range and in pursuit were far more common than other patterns of interaction. Because patterns of interaction were repeated, novel situations became rare, and the rate of learning was inversely proportional to time.

We now turn to Gerry. In this agent, when the KL determines an action, it submits that action to the PML, where actions have a one to one correspondence with behaviors. The PML commands have no direct counterparts in the KL. In Gerry, we assume that at the PML there are a finite number of actuators and corresponding commands. When the KL decides on action *A*, say *approach an object*, at the PML, a number of command transitions are constructed, one for each actuator command. Unlike Gabby, in which a command transition is built for each stimulus/response pair at the KL, in Gerry for each KL stimulus/response pair a set of command transitions is constructed.

In the following subsection, we discuss learning at the KL as a result of migration from the PML.

"Unconscious" to "Conscious" Migration

When a recurring pattern of interaction between an agent and its environment is detected at the "unconscious" level and this interaction is consistently beneficial for the agent in the sense that it always puts the agent in a more "desirable" state, a concept can be constructed at the "conscious" level to represent this pattern. We consider this type of migration from the PM-level to the KL a form of *skill acquisition* or *habit formation*.

To illustrate, a pattern of interaction at the PML consists of a sequence of commands, C (e.g., c_1; c_2, ..., c_n) and the situations, S (e.g., s_1; s_2; ...; s_n) for which these commands were generated. All commands in the sequence were generated while behavior B was in effect. B may or may not have a goal counterpart, G, at the KL. If it has, a new act, A, is created to stand for C, and the triple $<s_1, G, A>$ is migrated to the KL. At the KL, the knowledge that s_1 is the precondition of C is constructed. In case there is no corresponding goal in the KL, a primitive act and the situation s_1 that triggered the sequence of commands, C (i.e., the precondition for c_1;) is migrated. The situation, s_1, will play the role of a precondition for the act.[3] From this triple, knowledge constructs are created to reflect the fact that s_1 is the precondition of C. Also learned is the planning knowledge that every time G needs to be achieved, A should be performed.

Skill Refinement

Our premise for skill refinement is that we have an agent who has some basic skills but for certain situations can't choose the best alternative because the consequences of the actions are unknown and there are inadequate biases from experience. In skill refinement, we are concerned with "unconscious" choices of action. Often what is migrated from the KL to the PML introduces nondeterminacy (i.e., for certain situations, more than one action is applicable). In this subsection, we will look at how an agent can improve its skills by improving its choice of action.

The rules of Gabby's PMA are situation/command pairs where a situation is paired with multiple commands. The object of learning is to learn which command, when performed in a given situation yields, the better result (i.e., results in a situation that is more "desirable").

Some situations in ABS are more desirable for Gabby than others (e.g., being right behind the enemy and in shooting range). To each situation, S, we assign a reward value, R(S), between -1 and 1. As Gabby makes a move, it finds itself in a new situation. This new situation is not known to Gabby beforehand since the situation also depends on the other pilot's move. Since the new situation is not uniquely determined by Gabby's move, the game is not Markovian. The reinforcement-based learning we describe in this section has been proven to be effective only in Markovian environments.[4] Although we have successfully applied reinforcement learning in the non-Markovian domain of ABS, we cannot prove convergence of reinforcement learning in ABS.

In all of our modeling, we used Q-learning (Watkins 1989), which is a particular variety of reinforcement-based learning. In Q-learning, Utility(S(t), C(t)) is the evaluation of the appropriateness of command C 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 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.

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

The parameter γ determines how important it is to be in the state that Gabby ends up in after his move. In reinforcement based learning, this is known as the discount factor. Parameter λ is known as the recency/learning factor.

Example of Skill Refinement in the Agents — To test our implementation of reinforcement-based learning we tested Gabby in some playing situations. We developed a programmed opponent (a pseudo-human player) for Gabby

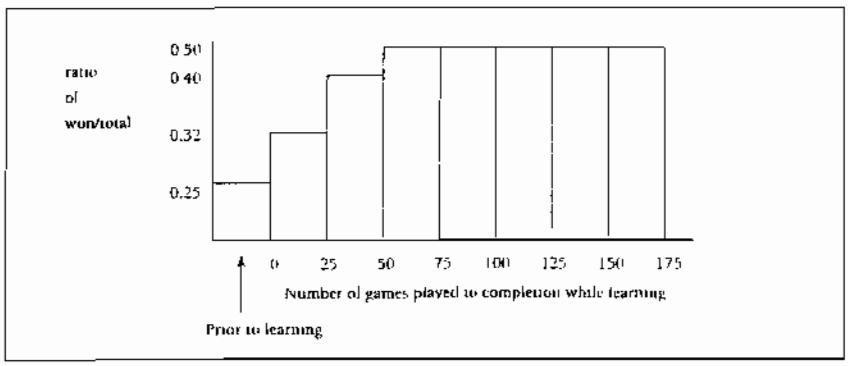

Figure 8 Learning rate using reinforcement based Q learning

that plays according to a fixed strategy written to simulate a strategy that a human player might use We had the two programs play 25 games without learning. Gabby won about 30% of these games. We then turned on Gabby's reinforcement based learning, had the programs play an additional 175 games, and recorded the number of winning games for each set of 25 Gabby won 32% of the first learning set, 40% of the second set, 50% of the third set, and remained at that level thereafter. Gabby learned to be just as good as its opponent in winning the game. Once one of the planes gets behind the other, it is very hard to shake off the enemy Gabby learned to be just as good as its enemy in finishing the enemy once in pursuit and attack. From our analysis of starting the game positions, it was equally probable for either plane to be in a position of pursuit and attack When one plane had the opportunity to be in pursuit, losing by the pursuant became highly likely since both pilots are equally competent. Given its opponents strategies, we conjecture that Gabby learned as much as possible. If we now were to improve the strategies of the pseudo-human player, we believe Gabby would improve again to match the level of competence of its opponent

We assume that Gabby does not "know" about the long-term consequences of its actions At the PML, we want to preserve reactivity and therefore, don't want the agent to learn long-term consequences of its actions We also do not want to rely completely on learning "unconsciously" through reinforcement. The agent must, therefore, observe interactions with the world in order to learn sequences of actions that are in no way guaranteed to be successful but that are commonplace for the agent and have been used frequently with a high degree of success. In the following subsection, we briefly discuss learning from routine activities in the world

"Routines" as Emergent Skills or Habits

"Routines are patterns of interaction between an agent and its world' (Agre

269). At times, these routines coincide with a *skill* or a *habit*. Our
n of a routine is "a course of action that is habitually (or by choice)
to." For simplicity, we can consider a routine as any frequently re-
equence of actions The adherence to a sequence of actions is a "de-
ollow one action after another without regard to applicability of ac-
nce by routine we imply a tendency to follow a sequence of actions
' at first glance this definition might seem to run counter to the situ-
view of activity (e.g., Clancey 1996) we have opted for in the PML
ast to traditional views, situated activity and situated cognition pro-
illow internal models of the world
ed activity replaces "thinking ahead" with experiences of interactions
world using the world as its own best model In our situated ap-
we want actions based on current situations, not based on reasoning
nsequences of actions. With a routine, we want actions performed in
' based merely on succession from previous actions. Such a course of
squarely in the spirit of situated theory.
l routines are noticed by the agent. An agent may engage in a routine
be "aware" of it. We call routines that the agent becomes "aware" of
routines We use the term emergent to mean unpredictable and nec-
arising from interaction between the agent and its world. For in-
when an agent observes a substantially improved situation over a
mber of actions, the agent may recognize a successful routine. We
all conscious routines with concepts available for the agent's reason-
e routines that reach the KL may be treated as a single action
: two ways emergent routines might be learned. We discussed one
arning emergent routines in the section on migration of knowledge
"unconscious" to the "conscious." A second technique is for the
'unconsciously" monitor the primitive actions at the KL that start off
'hen there is sufficient repetition in the sequence of KL actions, the
of actions is learned as a new complex action made up of the prim-
ons.
PML, routines are like programs that compete to gain control. Like
habit, once a routine gains control, it will usually run to completion
nterruption. However, if a routine is interrupted by a process in an-
t of the agent's architecture, something out of the ordinary (unex-
nust have happened in the world. This is reminiscent of classical
problems and recovery from failures (Georgeff 1987). However, we
iat with the emergence of situated cognition and the fact that reac-
ms make no predictions about the results of actions, the concept of
ilure will be avoided

Related Work

chitecture and approach has been an attempt at bridging the gap be- modeling deliberation (i.e., knowledge driven activity) and reactivity till driven activity). Our three layered architecture facilitates mediation n deliberation and reactivity. There are similar layered architectures pecific mediating layers (e.g., Bonasso *et al* 1992; Gat 1992) A system role of mediation is the Reactive Action Packages (RAPS) of Jim Firby RAPS takes a set of high-level tasks generated by a planner and re- ly decomposes them into a set of reactive skills that can be run on the RAPS then activates and deactivates these sets of skills in order to ac- sh tasks RAPS is also responsible for monitoring the execution of the o see if they are moving the robot towards the goal If the task is not achieved, RAPS can choose another method for achieving the same - take other corrective action

agents learn within a layer and migrate expertise in between layers A r learning scheme is described in Bonasso and Kortenkamp (1994). ng in the highest level improves cognitive skills Learning in the mid- er improves the coherence between the other levels Learning in the level improves the agents reflexive actions In our architecture, and so's, knowledge that is migrated from the highest level to lower levels . a task that is highly cognitive in the beginning (i.e , cognitive exper- reactive and skilled in the future. In our architecture we go one step r and allow for repeated patterns of interactions (i.e., motor skills) at w level to be redefined using cognitive concepts at a later time.

Summary and Conclusion

pproach to the study of expertise is to build agents that use and acquire ledge and skill. Our architecture for intelligent agency distinguishes be- ; "conscious" and "unconscious" processes Our approach involves the diment of actions in that it addresses migration of knowledge and the sition of new skills that emerge from repetition

used our architecture in modeling agents in several domains. One of domains is a video-game in which a computer agent flies a WWI-style ne As the game progresses, this agent learns to maneuver the airplane "unconsciously" and improves its decision making ability This im- ment shows a style of building skills from the agent's own knowledge ts interactions in the world Our capturing skills are rather novel, espe- among advocates of symbolic AI

: have developed some preliminary techniques for learning routines

an agent's interactions with the world. By learning routines, an agent ›ecome more automatic in its interaction in the world, and with certain nes, it can enrich its "conscious" knowledge about actions

Acknowledgment

work was supported in part by Equipment Grant No EDUD-US-)22 from SUN Microsystems Computer Corporation. We are grateful to ı Lammens and William Rapaport for their comments on the issues dis-:d in this chapter. The first author thanks Jeff Shrager of Xerox PARC for ıl comments We are grateful to editors Ken Ford, Paul Feltovich, and rt Hoffman for many comments that improved our chapter

Notes

ıs is a highly simplistic model of perception and action. For a more detailed the-l perception and action, see Kelso (1985)

ır distinction between acts and behaviors and the levels we choose to place them .AIR are consistent with "Acts, as I use the term, are what humans do when they d to do something, whereas behavior is unintended" (Collins 1990, p 30). As Collins, we see intentionality to be the distinguishing characteristic of acts and ›iors

: will either have all the commands as acts at the KL or just invent a new act for :quence of commands, but the commands themselves will not exist at the KL.

ıe Markovian assumption for an agent is that the effect of an agent's actions only ıds on its current state

References

P.E. & Chapman, D. (1987) Pengi: An implementation of a theory of activity. In *edings of American Association for Artificial Intelligence* (pp. 268-272). Seattle, WA Press

·rson, J (1983) *The Architecture of Cognition*. Cambridge Harvard University

sso, R.P., Antonisse, H.J & Slack, M.G (1992). A reactive robot system for find ·etch tasks in an outdoor environment In *Proceedings of the Tenth National Confer-m Artificial Intelligence* (pp 801-808). Menlo Park, CA: AAAI Press.

sso, R.P & Kortenkamp, D. (1994). An intelligent agent architecture in which to ıe robot learning In *Working Notes. MCL COLT 1994 Robot Learning Workshop.*

:ey, W.J (1996). The conceptual nature of knowledge, situations, and activity. In eltovich, K.M. Ford & R.R Hoffman (Eds.), *Expertise in Context* (pp 247-291) o Park, CA: AAAI/MIT Press

ns, H. (1990) *Artificial Experts: Social Knowledge and Intelligent Machines* Cam-ge, MA. The MIT Press

: R.J (1992). Building symbolic primitives with continuous control routines. In ·edings of the International Conference on AI Planning Systems

Gat, E (1992) Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots In *Proceedings Tenth National Conference on Artificial Intelligence* (pp 809-815) Menlo Park, CA: AAAI Press

Georgeff, M.P. (1987) Planning. In *Annual Reviews of Computer Science*, Vol 2, (pp 359-400). Palo Alto, CA Annual Reviews.

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

Hexmoor, H , Lammens, J & Shapiro, S (1992) *An autonomous agent architecture for integrating perception and acting with grounded, embodied symbolic reasoning* (Tech Rep CS-92-21) Dept of Computer Science, SUNY at Buffalo, NY

Hexmoor, H , Lammens, J & Shapiro, S.C (1993) Embodiment in GLAIR A grounded layered architecture with integrated reasoning In D.D Dankel (Ed.), *Proceedings of the Sixth Florida Artificial Intelligence Research Symposium* (pp 325-329) [Also available as CS Dept TR93-10. SUNY at Buffalo, NY]

Hexmoor, H & Nute, D (1992) *Methods for deciding what to do next and learning* (Tech Rep. AI-1992-01). AI Programs, University of Georgia, Athens, GA. [Also available as CS Department TR-92-23 SUNY at Buffalo, NY.]

Kelso, J.A.S (Ed) (1985). *Human Motor Behavior* Hillsdale, NJ Erlbaum

Laird, J.E., Newell, A & Rosenbloom, P.S (1987). Soar: An architecture for general intelligence *Artificial Intelligence Journal, 33,* 1-64

Kumar, D & Shapiro, S.C (1991) Modeling a rational cognitive agent in SNePS In P Barahona, L M Pereira & A Porto (Eds), *EPIA 91 5th Portugese Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence,* 541 (pp 120-134) Heidelberg. Springer-Verlag.

Miller G (1956). The magical number seven, plus or minus two Some limits on our capacity for processing information. *Psychological Review, 63-2,* 81-97

Shapiro, S C & Group, T.S.I. (1992) *SNePS-2 User's Manual* Dept. of Computer Science, SUNY at Buffalo, NY

Shapiro. S.C & Rapaport, W.J (1992) The SNePS family. *Computers & Mathematics with Applications, 23(2-5),* 243-275

Shiffrin, R.M. & Schneider, W (1977) Controlled and automatic human information processing. II. perceptual learning, automatic attending, and a general theory. *Psychological Review, 84-2,* 127-190

Sutton, R (1988) Learning to predict by the methods of temporal differences *Machine Learning, 3,* 3-44

Taylor, D (1991) *Bowling Strikes* Chicago. Contemporary Books.

Watkins C (1989). *Learning from delayed rewards* Ph D. thesis, Kings College. Cambridge, UK