This is a slightly revised version of a paper that will appear in the proceedings of AI In Engineering, Toulouse, France, June 29 - July 1, 1993.

Behavior Based AI, Cognitive Processes, and Emergent Behaviors in Autonomous Agents

Henry Hexmoor, Johan Lammens, Guido Caicedo, and Stuart C. Shapiro
Department of Computer Science
226 Bell Hall
State University of New York at Buffalo
Buffalo, NY 14260

ABSTRACT

Behavior based AI [Brooks, 1990, Maes, 1990] has questioned the need for modeling intelligent agency using generalized cognitive modules for perception and behavior generation. Behavior based AI has demonstrated successful interactions in unpredictable environments in the mobile robot domain [Brooks, 1985, Brooks, 1990]. This has created a gulf between "traditional" approaches to modeling intelligent agency and behavior based approaches. We present an architecture for intelligent autonomous agents which we call GLAIR (Grounded Layered Architecture with Integrated Reasoning) [Hexmoor et al., 1992, Hexmoor et al., 1993b, Hexmoor et al., 1993a]. GLAIR is a general multi-level architecture for autonomous cognitive agents with integrated sensory and motor capabilities. GLAIR offers an "unconscious" layer for modeling tasks that exhibit a close affinity between sensing and acting, i.e., behavior based AI modules, and a "conscious" layer for modeling tasks that exhibit delays between sensing and acting. GLAIR provides learning mechanisms that allow for autonomous agents to learn emergent behaviors and add it to their repertoire of behaviors. In this paper we will describe the principles of GLAIR and systems we have developed that demonstrate how GLAIR based agents acquire and exhibit a repertoire of behaviors at different cognitive levels.

1 Overview of GLAIR

What goes into an architecture for an autonomous agent has traditionally depended to a large extent on whether we are *building a physical system*, *understanding/modeling behaviors of an anthropomorphic agent*, or *integrating a select number of intelligent behaviors*. The organization of an architecture is also influenced by adopting various philosophical positions like Fodor's *modularity* assumption [Fodor, 1983], or a *connectionist* point of view, e.g. [McClelland et al., 1986], or an *anti-modularity* 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 [Chapman, 1990]. Our architecture is characterized by a three-level organization into a Knowledge Level (KL), a Perceptuo-Motor Level (PML), and a Sensori-Actuator Level (SAL). 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, i.e. a *behavior-based* architecture. The most important difference from a purely 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 KL. Representation, reasoning (including planning), perception, and generation of behavior are distributed through all three levels. Our architecture is best described using a resolution pyramid metaphor as used in computer vision work [Ballard and Brown, 1982], rather than a central vs. peripheral metaphor.

Architectures for building physical systems, e.g. robotic architectures [Albus et al., 1981], tend to address the relationship between a physical entity like a robot and 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 architectures address what John Pollock calls *Quick and Inflexible* (Q&I) processes [Pollock, 1989].

We define consciousness for a robotic agent operationally as being aware of one's 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, and (3) being able to communicate with an external agent about the environment.

Architectures for understanding/modeling behaviors of an anthropomorphic agent, e.g., cognitive architectures [Anderson, 1983, Pollock, 1989, Langley et al., 1991], tend to address the relationships that exist among the structure of 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, and the other way around.

We are interested in autonomous agents that are embedded in a dynamic environment. Such an agent needs to continually interact 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 paper, we will present a family of agents based on our architecture. These agents are designed with a robot in mind, but their structure is also akin to anthropomorphic agents. Figure 1 schematically presents our architecture.

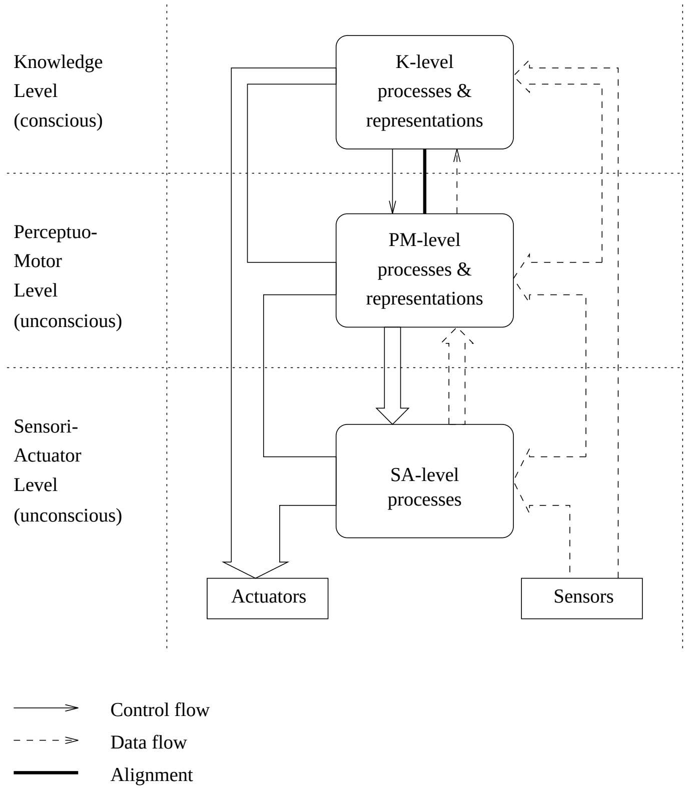

Figure 1: Schematic representation of the agent architecture. Width of control and data paths suggests the amount of information passing through (bandwidth). Sensors include both world-sensors and proprio-sensors.

There are several features that contribute to the robustness of our architecture. We highlight them below. For an in-depth discussion and comparison with other architectures see [Hexmoor et al., 1992].

- We differentiate conscious reasoning from unconscious Perceptuo-motor and sensori-actuator processing.
- The levels of our architecture are semi-autonomous and processed in parallel.
- 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 layered and not exclusively top-down.
- Lower level mechanisms can pre-empt higher level ones. This is kind of subsumption on its head, but everything depends on the placement of

behaviors in the hierarchy of course. We haven't quite decided yet whether inhibition should work the other way around as well.

- There is a correspondence between terms in the Knowledge Representation and Reasoning (KRR) system on one hand, and perceived objects, properties, events, and states of affairs in the world, and motor capabilities on the other hand. We call this correspondence *alignment*.
- The level at which any given behavior is generated and/or controlled is not fixed, but can vary in the course of learning, or depending on the particular goals and capabilities of the agent in question.

A major objective for GLAIR is learning emergent behaviors. Like Agre's improvised actions [Agre and Chapman, 1987] and Brooks's subsumption [Brooks, 1985] we believe complex behaviors emerge from interaction of the agent with its environment without planning. However, previous work in this area hard-coded primitive actions and did not attempt to learn the improvised behavior.

2 Consciousness

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, as it 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. 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 experimental investigations in this respect. The real test for our architecture is its usefulness in applications to physical (robotic) autonomous agents.

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. The implementation mechanisms we use for the unconscious levels, such as PMA, 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 (section 6.2), uses distinct hardware for the three levels.

2.1 Perceptuo-Motor Automata

At the perceptuo-motor level, the behaviors resulting in physical actions are generated by an automaton, which we will call a PM-automaton (PMA) [Hexmoor and Nute, 1992]. In other words, PMA are representation mechanisms for generating behaviors at an "unconscious" level. PMA is a family of finite state machines represented by <Rewards, Goal-Transitions, Goals, Action-Transitions, Actions, Sensations>. Goals, Rewards, and Goal-Transitions in a PMA can be empty. 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.

- < NIL, NIL, NIL, Action-Transitions, Actions, Sensations>
- <NIL, Goal-Transitions, Goals, Action-Transitions, Actions, Sensations>
- <Rewards, NIL, NIL, Action-Transitions, Actions, Sensations>
- <Rewards, Goal-Transitions, Goals, Action-Transitions, Actions, Sensations>

Actions are a set of primitive actions. Sensations are a set of atomic (i.e., nondecomposable) percept types in a PMA. For example, a percept type for a mobile robot that walks in building hallways can be *its distance to the wall.* In Air Battle Simulation (ABS), *relative distances from the enemy in X, Y , and Z axes* are percepts. We assume that all possible situations for a PMA can be specified by one or many Sensations, i.e., percept types. We call a pattern of Sensation types that are used for input to a PMA a Situation. In ABS, *<distanceX, distanceY, distanceZ, orientation>* is a Situation. Henceforth, we will refer to values of a situation input to a PMA as Situation instances. In ABS, *<close-front, close-right, close-above, parallel>* is a situation instance. Goals are descriptions of desirable conditions for a PMA. Goals are used to partition a PMA. Rewards are static goodness values associated with situations. These are determined apriori to PMA execution and remain unchanged throughout.

Action-Transitions (AT) are transformations that generate behaviors for execution, (see Figure 2). The simplest ATs are mappings of Sensations $\mapsto$ Action. Below is a list of AT types. ATs can be disabled by inhibiting them. This is useful when the agent wants to override unconscious behaviors generated by PMA with conscious behaviors generated by the knowledge layer.

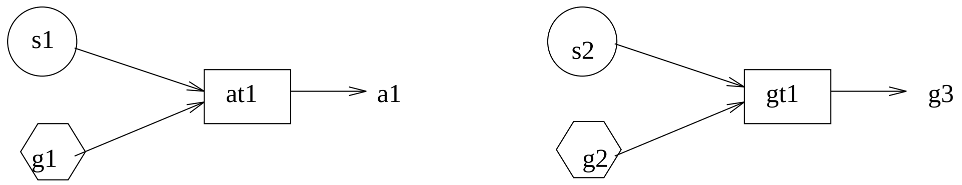

Figure 2: An action transition is shown on the left and a goal transition is shown in the right.

- Sensations $\mapsto$ Action

- Previous action $\times$ Sensations $\mapsto$ Action
- Goal $\times$ Sensations $\mapsto$ Action
- Tendency $\times$ Sensations $\mapsto$ Action
- Previous action $\times$ Goal $\times$ Sensations $\mapsto$ Action
- Previous action $\times$ Goal $\times$ Tendency $\times$ Sensations $\mapsto$ Action
- Tendency $\times$ Goal $\times$ Sensations $\mapsto$ Action

ATs may consider goals as antecedents. ATs may consider previous actions in deciding what to do next. We may take into account an estimated accumulated reward for actions. We call the latter, tendencies. Tendencies are computed using reinforcement based learning techniques [Sutton, 1988].

Goal-Transitions (GT) are transformations that update goals when the current goal is satisfied. GT is Goal1 $\times$ Sensations $\mapsto$ Goal2 where Goal1 and Goal2 are goals.

The PMA maintains the current goal. When the latest sensations along with the current goal match an action transition, that action transition activates an action which is then executed.

The primary mode of acquiring a PMA is intended to be by converting plans in the knowledge level into a PMA by a process described in [Hexmoor and Nute, 1992]. In the following section, we describe a stepwise learning scheme for acquiring PMA transitions.

3 Knowledge Migration from "conscious" to "unconscious" Level

Knowledge in GLAIR can migrate from conscious to unconscious levels. In our application Air Battle Simulation (ABS) described later in this paper (see [Hexmoor et al., 1993a] for detals of this system) we show how a video-game playing agent learns how to dynamically "compile" a game playing strategy that is initially formulated as explicit reasoning rules at the Knowledge level into an implicit form of knowledge at the Perceptuo-Motor level, a PMA.

When the knowledge level determines an action for execution, it reaches the PM-level. At the PM-level, this knowledge is learned in the form of PMA transitions. The idea is that the next time circumstances make this action applicable, it can be selected for execution without having to resort to "conscious" processes.

4 Learning Tendencies

Reinforcement based learning [Watkins, 1989] is a successful technique that is used for learning action consequences, also known as action models. In learning action models, we assume a Markovian environment. That is, the agent believes the world changes only due to its own actions. In contrast to learning action models, we are interested in modeling behavior generation by agents that function in dynamic environments. The impact of the agent's action for the agent depends on the situations in which actions are applied and on other agents' actions. Other agents' actions are nondeterministic. Furthermore, we assume that the agent is computationally and cognitively resource bound. We assume that the agent needs time to think about the best action and in general there is not enough time. In such an environment, we want the agent to observe interactions with the world in order to learn what actions tend to be successful in light of its goals and prevailing situations. This is useful when the agent has to arbitrate among multiple applicable actions.

Given a PMA action transition with multiple actions, each action is given a goodness value. After an action is executed, the reward for the agent is a function of the agent's own action plus whatever other agents have done to result in the new situation. Using reinforcement based learning, the goodness values are updated to reflect rewards received and predictions about goodness of resulting actions. However, unlike action models, convergence to optimal action selection is not guaranteed. Instead, applicability of actions can only be said to follow tendencies to act, i.e., what has worked in the past. We are experimenting with two variations of learning tendencies. The first is to assess rewards for each new situation reached after executing actions. This is somewhat unrealistic in that often the problem space is large and not completely understood (here we treat situations as problem states). Providing rewards in each situation is equivalent to providing a large body of knowledge about the environment. In the second variation, only a select number of situations are known to be significant so there is a reward for reaching that situation. Most other states are insignificant. In this variation, goodness values of actions are updated only when a significant situation is reached.

5 Learning Emergent Routines

We endow the agent with a minimal number of primitive actions and sensations. Our basis for this minimality and choice of primitive actions is physiological. In other words, in our modeling a computer agent, we will choose actions that are physically basic for the robot as primitive actions for the computer agent. We then instruct the agent to perform tasks and in the midst of accomplishing them, we expect the agent to notice basic behaviors (i.e., routines) emerge. An example of an emergent behavior we explore is when a mobile robot learns *moving toward an object*. Emergent behaviors are learned and consequently used in the agent's repertoire of actions.

We assume that the agent does not know about long term consequences of its actions. Over a finite number of actions, when the agent observes a substantially improved situation, chances are he has found a successful *Routine*. We record such detected *Routines* and as they recur, we increase our confidence in them. When our confidence in a *Routine* reaches a certain level, a concept is created at the Knowledge level of GLAIR for the routine and from then on, this routine can be treated as a single action at that level, [Hexmoor, 1992]. Learning routines is closely related to the second variation of learning tendencies as we discussed in the previous section. Instead of learning action goodnesses, to learn a routine we record the sequence of actions between a significantly bad situations and a significantly good situation.

6 Applications of GLAIR

6.1 A simulation study: Air Battle

We have written a program, Air Battle Simulation (ABS) [Hexmoor et al., 1993a], that simulates World War I style airplane dog-fights. ABS is an interactive video-game in which a human player plays against a computer-driven agent. The game starts up by displaying a game window and a control panel window (Figure 3). The human player's plane is always displayed in the center of the screen. The aerial two-dimensional position of the enemy plane is displayed on the screen with the direction of flight relative to the human player's plane. The human player looks at the game screen to determine his airplane's position and orientation with respect to the enemy's plane. (S)he then uses the control panel to choose a move. A move is a combination of changing altitude, speed, and direction. When the human player presses the go button, the computer agent also selects a move. The game simulator then considers the human player's move and the computer agent's move to determine the outcome of moves, and updates the screen and the accumulated damage to planes. ABS simulates simultaneous moves this way. If a player's plane is close in altitude and position to the enemy plane, and the enemy is in frontal sight, the latter is fired on automatically (i.e., firing is not a separate action). The levels of damage are recorded in a side panel, and the game ends when one or both of the two player's planes are destroyed.

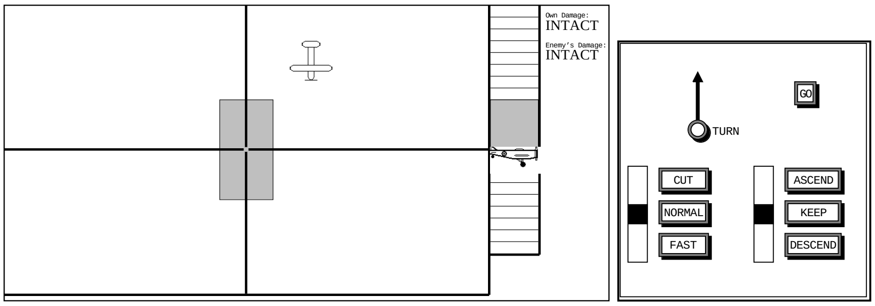

Figure 3: Air Battle Simulation game window and control panel (see text).

Initially, the agent has not acquired a PMA, and uses conscious level reasoning to decide what move to make. Once transitions are learned and cached in a PMA, the agent uses the PMA for deciding its next move whenever possible. By adding learning strategies, a PMA is developed that caches moves decided at the Knowledge level for future use. Learning can be used to mark PMA moves that prove unwise and to reinforce moves that turn out to be successful. We started ABS with an empty PMA and as the game was played, transitions of the PMA were learned. Also as the transitions were learned, when similar situations occurred and there was an appropriate PMA response, the PMA executed that action. As the game was played, we observed that the agent became more reactive since the PMA was increasingly used to generate behaviors instead of the Knowledge level.

We are exeprimenting with reinforcement based learning, emergent routines, and other learning techniques such as experimentation as a form of learning [Shen, 1989]. We are also interested in developing experiments that will help in psychological validation of GLAIR and the learning strategies used in ABS. As of this writing ABS is fully operational.

6.2 A physical implementation: the Robot Waiter

We are developing an experimental setup in which a robot arm will set a dinner table in various configurations, guided by input from a color camera and controlled by a host computer.

The physical setup includes a dinner table, a supplies table containing kitchenware, a Puma 260 robot arm, a CCD camera, a PC-based color frame grabber, and a Sun 4/260 workstation host computer. In a later phase of this project, we hope to replace the Puma 260 robot arm with a larger Puma 560 model. Figure 4 represents the setup.

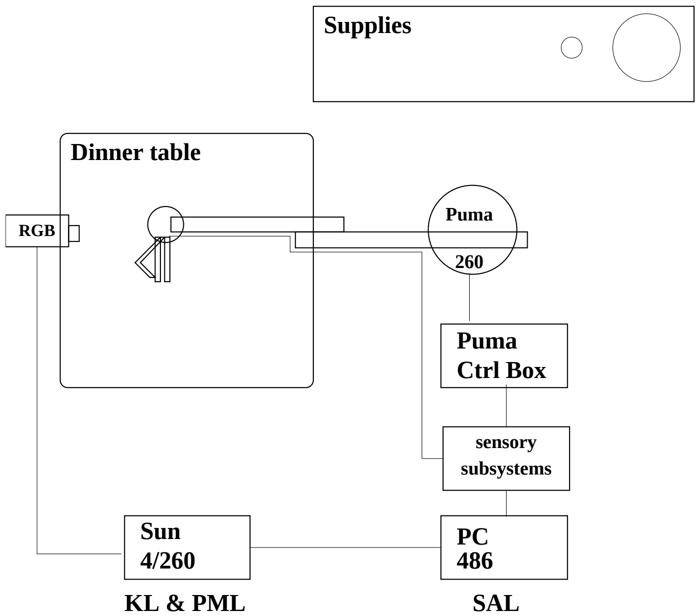

Figure 4: The Robot Waiter physical setup.

The human operator instructs the agent to set the table for dinner, breakfast, etc., specifying by named color which objects to choose from the supplies table (color is not a sufficient feature to recognize objects, as each type of object is available in several colors). The camera is mounted in a fixed position overlooking both tables and the robot. This testbed provides a dynamic, yet controllable, environment in which an agent is placed so as to facilitate empirical studies of its behavior. The places, number, kind, and color of objects is not fixed, and unannounced human intervention in the domain is allowed while the robot is carrying out its task.

We call the agent for this project the Robot Waiter (RW). RW is being developed in accordance with the principles of the GLAIR architecture. Figure 5 schematically presents its structure. The categorizer uses domain constraints to determine what objects are in the visual field. It can also be told to look for a certain object, e.g., *a red cup*. The sensory memory acts as an attentional register, keeping track of the object that is being manipulated or is to be inspected.

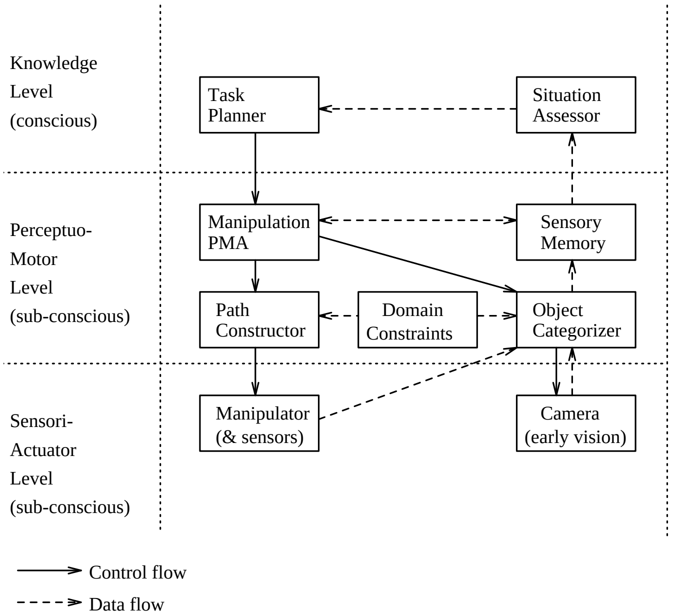

Figure 5: Schematic representation of the Robot Waiter GLAIR-agent.

Actions are transmitted from KL to PML, e.g., *look for a red cup* and *pick it up*. Once the sensory memory contains an object matching the object desired, actions involving that object can be understood at the PML. For instance, once a red cup is discovered and recorded in the sensory memory, an action like *pick it up* is expanded by a PMA into robot arm tasks to reach for the cup, grasp it, and lift it. A PMA is a implementation mechanism for routine activities at an "unconscious" level, [Hexmoor and Nute, 1992, Hexmoor et al., 1992, Hexmoor et al., 1993a]. Each task involving a robot motion is subsequently submitted to the path constructor. Some motions may have to be decomposed to visit intermediate points in order to avoid fixtures in the environment. The path constructor generates path segments for each robot motion. Each path segment generated by the path constructor is transmitted to the SAL for execution.

RW incorporates an embodied model of color perception and color naming, modeled after the physiology of human color perception. This model allows the agent to (1) name colors shown to it, and express a typicality judgement, (2) point out examples of named colors in its environment, and (3) learn new names for colors. This model provides the perceptual grounding for a set of basic color terms [Berlin and Kay, 1969] represented at the Knowledge level. The color domain was chosen as a case study for embodied perception because of the relative abundance of psychological, psychophysical and neurophysiological data in the literature [Lammens,]. It is a complex enough domain to allow the usefulness of embodiment for computational models of perception to be demonstrated, yet feasible enough to be implemented in actual autonomous agents. As of this writing, the Robot Waiter project is partially implemented.

6.3 A simulation study: the Mobile Robot Lab

The Mobile Robot Lab (MRL) is a simulation environment. The simulation is relatively simple, but nevertheless provides a rich and realistic enough environment to function as a testbed for the development of physical GLAIR-agents. A complete setup using MRL consists of a GLAIR-agent, a simulator with an incorporated description of a physical environment, and a graphical interface (Figure 6). 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 power distribution (SPD).

Any number of robots may inhabit the room. They have two independently driven wheels on either side, and two small support wheels underneath in the front and the back. Furthermore a bumper bar front and back, with contact and force sensors built in, and a color camera on top, parallel to the direction of the driven wheels. The camera is fixed and mounted horizontally. The robot also has a non-directional light with a user-defined SPD on top, which it can switch on and off or flash.

The simulator interfaces with the agent and with the graphical interface. It takes care of any I/O with the agent that would otherwise come from the sensors and go to the actuators of a real mobile robot. It also takes care of any I/O with the graphical interface, needed to keep the graphical display of the robot and its physical environment updated.

The simulator incorporates a simplified model of the physics of motion and sensing for the mobile robot. It continually updates the position of the robot depending on the rotation speed and direction of its wheels, and provides the agent with appropriate sensory data about wheel rotation and contact with objects. The simulator provides simulated camera input to the agent. Camera input is simplified in that it consists of a 9x7 pixel array (square pixels), with each pixel represented as an RGB triplet. This simplified camera view is computed and passed to the simulator by the graphical interface, on the basis of the 3D perspective views (see below).

The graphical interface provides a real-time view of the robot and the robot's environment, using data obtained from the simulator. It consists of a 2D display showing a bird's eye view of the room, the objects, and the robot in it, a sensors and actuators monitor display, and a 3D perspective display that shows the environment from the robot's point of view.

The agent's primitive actions are limited to independently controlling the

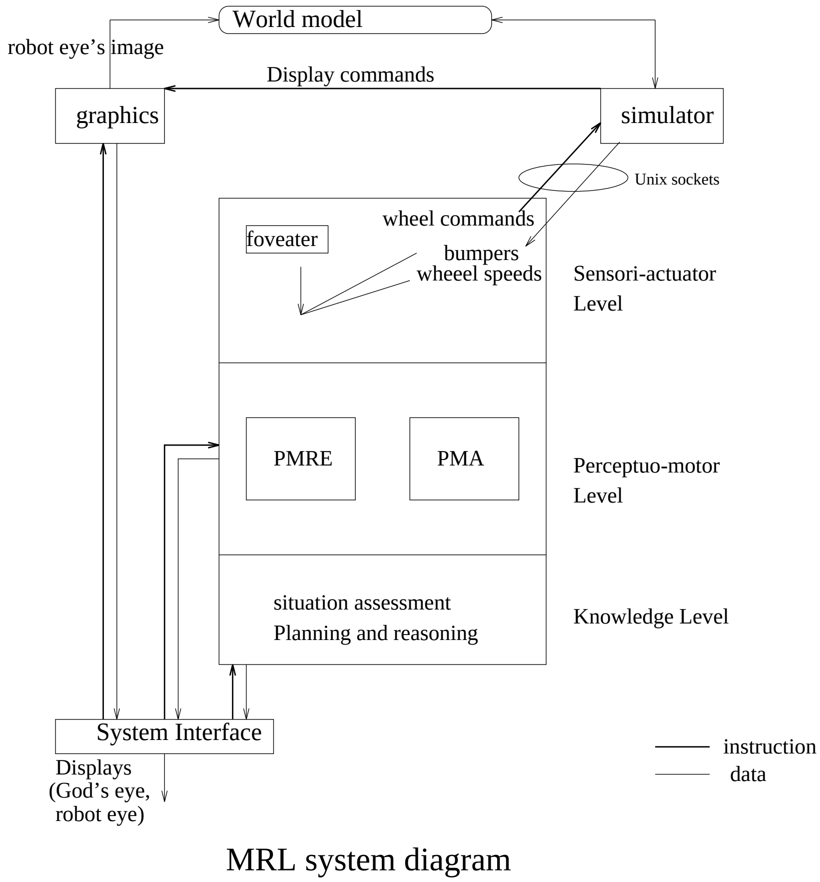

Figure 6: MRL system diagram. It consists of a GLAIR-agent, a simulator with an incorporated model of a physical environment, and a graphical interface. Arrows represent direction of data flow among the components.

speed and direction of rotation for each of the left and right wheel motors. Actuation values for each motor increase/decrease by 1/10 foot/sec of wheel circumference in forward and reverse direction. Brakes can be applied to each wheel. We assume negligible acceleration/deceleration times.

As shown in Figure 6, the agent receives an image, bumper information, and wheel speeds at the SA level. The image is subsequently foveated [Bandera and Scott, 1989] to model foveation after advanced biological vision. (Foveation produces high resolution at the center of the image and decreasing resolution at the image periphery.) After some early vision processing, the processed image data in the form of blobs as well as other sensory information reaches the Perceptuo-Motor Reduction Engine (PMRE) which in turn assesses the situation perceived. This involves vision processing such as looming effects as well as fusion of multiple sensory data. PMRE feeds PMA with the current situation for generating learned reactions. Assessed situations reach the knowledge level which may do further processing on the sensory data or use that information to generated instructions for execution.

When the robot is in motion, it will use cues from its environment to guide its behavior selections and subsequent learning. In order for the robot to guide its behavior, we need to associate rewards with actions that result in desirable sensations. For instance, if the robot wants (at the Knowledge level) to touch an object and is taking actions to move towards the object, and the object is in the left field of view and the robot moves left (increases its right wheel motor speed) it will bring the object to the center of the field of view. The action of turning left in this situation will be positively rewarded. The result of learning (sequences of) actions will be recorded in PMA transitions. For instance, touching an object evolves into a set of PMA transitions.

At the knowledge level, for each behavior, a triple of <A, S, R> will be defined. A is the set of primitive actions, S is a set of sensations, and R is a set of rewards. For example, for the behavior of touching, A = {left wheel forward increase speed, left wheel forward decrease speed, right wheel forward increase speed, right wheel forward decrease speed}; S = {object is bigger, object is smaller, object is in the left field of view, object is in the right field of view, object is in the center of the field of view, object is too close, contact is made}; R = {object is bigger +1, object is smaller -1, object is in the left field of view -1, object is in the right field of view -1, object is in the center of field of view +1, object is too close +1, contact is made +1} (numbers ranging from +1 to -1 are rewards with +1 denoting desirable and -1 denoting undesirable).

Below we give a list of emergent behaviors that the robot learns completely on its own.

- Touch behavior: approach an object until contact is made.
- Proximity percepts: proximity to an object, derived from camera data.
- Approach behavior: approach an object until it is in proximity.
- Block percept: recognizing something in the field of view as a block.
- Push a block
- Find a block
- Unwedge a block
- Explore/map the room

At the Knowledge level, all of the percepts and behaviors of the Perceptuo-Motor level are represented, but in a more symbolic fashion. For instance, colors and shapes have names. The representations at the two levels are connected via the alignment mechanism discussed above. Also at this level is a symbolic map of the room and the objects in it, and the current position of the agent. In general, planning and some learning activities can originate at this level, and reasoning about the environment and the agent's actions, perceptions, goals, desires, states, etc. is confined to this level only. Concepts of space and time would also be represented at this level, perhaps as emergent concepts from the behavior of the agent in its environment.

7 Summary and Conclusion

We have proposed an architecture that models "conscious" and "unconscious" processes for behavior generation and learning by an intelligent agent. We distinguish three levels in our architecture: Knowledge Level, Perceptuo-Motor Level, and Sensori-Actuator Level. Generation and control of behaviors is distributed over all levels, and each level has independent access to sensors (via perceptual reduction) and (to some extent) actuators. As we move down the levels, computational and representational power is traded off for better response time and simplicity of control. GLAIR agents learn from their interactions with the environment. A video-game agent, a mobile robot simulator, and a robotic agent are developed that demonstrate the principles used in GLAIR. These agents demonstrate emergent behaviors and integration of perception and action in the architecture.

References

[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.

[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.

[Anderson, 1983] Anderson, J. R. (1983). *The Architecture of Cognition.* Cambridge: Harvard University Press.

[Ballard and Brown, 1982] Ballard, D. H. and Brown, C. M. (1982). *Computer Vision.* Prentice-Hall, Englewood Cliffs, NJ.

[Bandera and Scott, 1989] Bandera, C. and Scott, P. (1989). Foveal machine vision systems. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 596–599.

[Berlin and Kay, 1969] Berlin, B. and Kay, P. (1991 (orig. 1969)). *Basic Color Terms: Their Universality and Evolution.* University of California Press, Berkeley CA, first paperback edition.

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

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

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

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

[Hexmoor, 1992] Hexmoor, H. (1992). Representing and learning successful routine activities. Technical Report Unpublished PhD Proposal, Dept. of Computer Science, SUNY at Buffalo, New York.

[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*. Dept. of Computer Science, SUNY at Buffalo, New York.

[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., 1993b] Hexmoor, H., Lammens, J., and Shapiro, S. C. (1993b). Embodiment in GLAIR: A Grounded Layered Architecture with Integrated Reasoning. In *Florida AI Research Symposium*.

[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.

[Lammens,] Lammens, J. M. A computational model of color perception and color naming: a case study of symbol grounding for natural language semantics. Dissertation proposal, SUNY/Buffalo CS department, June 1992.

[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.

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

[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.

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

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

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

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