Of Elephants and Men *†

Johan M. Lammens, Henry H. Hexmoor, Stuart C. Shapiro

April 28, 1993

Autonomous Agents Lab
Computer Science Department
State University of New York at Buffalo
Buffalo, NY 14260
tel: (716) 645-3180, fax: (716) 645-3464
e-mail: lammens@cs.buffalo.edu

To appear in proceedings of the NATO-ASI on the Biology and Technology of Intelligent Autonomous Agents, Trento, Italy, March 1–12 1993.
(preliminary version)

Abstract

In the elephant paper [Bro90], Brooks criticized the ungroundedness of traditional symbol systems, and proposed physically grounded systems as an alternative, in particular the subsumption architecture. Although we are still struggling with many of the issues involved, we believe we have some contributions to make towards solving some of the open problems with physically grounded systems, particularly with respect to whether or how to integrate the old with the new. In this paper we describe an agent architecture that specifies an integration of explicit representation and reasoning mechanisms, embodied semantics through grounding symbols in perception and action, and implicit representations of special-purpose mechanisms of sensory processing, perception, and motor control. We then present components that we place in our general architecture to build agents that exhibit situated activity and learning, and finally a physical agent implementation and two simulation studies. The gist of our paper is that the Brooksian behavior generation approach goes a long way towards modeling elephant behavior, which we find very interesting, but that in order to model more deliberative behavior we may also need something else.

*This work was supported in part by Equipment Grant No. EDUD-US-932022 from SUN Microsystems Computer Corporation.

†Apologies to John Steinbeck, who carries no blame for this paper.

Contents

1 Introduction and Overview

In the elephant paper [Bro90] appearing in the proceedings of the predecessor of the current workshop, 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 the current paper we explore the possibilities for 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 we are still struggling with many of the issues involved, we believe we can contribute to a solution for some of the problems for both classical systems and physically grounded systems mentioned in [Bro90], in particular:

- The ungroundedness of symbolic systems (referred to as "the symbol grounding problem" by [Har90]): 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 we use in our implementations allows the use of unnamed intensional concepts.
- We have some ideas about how to automate the construction of behavior generating modules through learning, but much remains to be done.

We agree with the requirement of physically implemented systems as the true test for any autonomous agent architecture, and to this end we are working on several different implementations. We will present both our general multi-level architecture for intelligent autonomous agents with integrated sensory and motor capabilities, GLAIR[1], and a physical implementation and two simulation studies of GLAIR-agents.

By an *architecture* we mean an organization of components of a system, what is integral to the system, and how the various components interact.[2] 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/modeling behaviors of an anthropomorphic agent*, or *integrating a select number of behaviors*. The organization of an architecture may also be influenced by whether or not one adopts the *modularity* assumption of Fodor [Fod83], or a *connectionist* point of view, e.g. [MRH86], or an *anti-modularity* assumption as in Brooks's subsumption architecture [Bro85]. 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 [Cha90]. 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, i.e., a *behavior-based* architecture. The most important difference with a behavior-based architecture like Brooks's subsumption 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. Our architecture is best described using a resolution pyramid metaphor as used in computer vision work [BB82], rather than a central vs. peripheral metaphor.

Architectures for building physical systems, e.g., robotic architectures [ABN81], 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. We say these systems are

[1] Grounded Layered Architecture with Integrated Reasoning

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

not concerned with *consciousness*. These architectures address what John Pollock calls *Quick and Inflexible* (Q&I) processes [Pol89]. 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.[3]

Architectures for understanding/modeling behaviors of an anthropomorphic agent, e.g., cognitive architectures [And83, Pol89, LMA91], 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[4] 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.

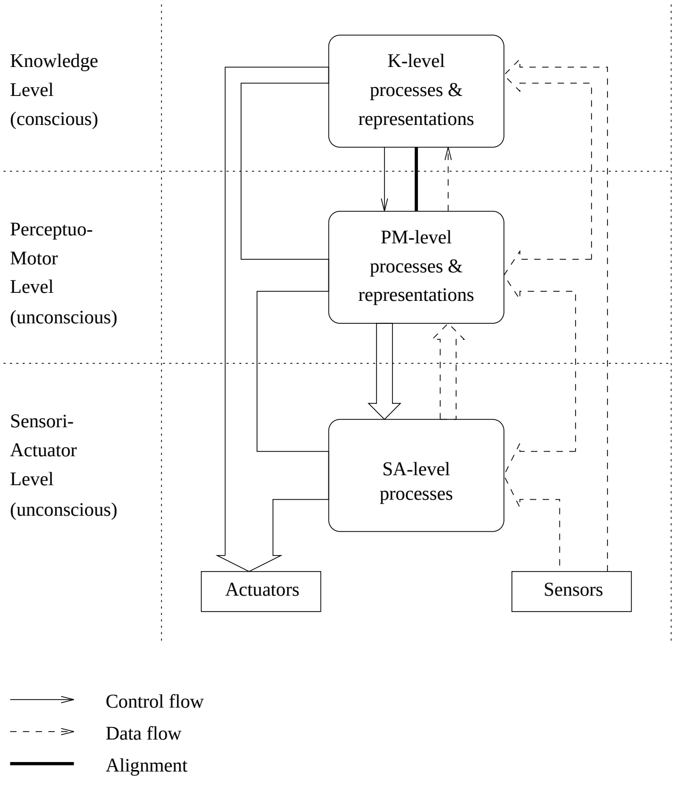

Figure 1: Schematic representation of the agent architecture. Width of control and data paths suggests the amount of information passing through (bandwidth). Sensors include both world-sensors and proprio-sensors.

[3] A machine like a vending machine or an industrial robot has responses, but it is *unconscious*. See [Cul63] for a discussion of independence of consciousness from having a response. Also, intelligent behavior is independent of consciousness in our opinion.

[4] "Embedded agents are computer systems that sense and act on their environment, monitoring complex dynamic conditions and affecting the environment in goal-oriented ways." ([KR90] page 1).

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

- We differentiate conscious reasoning from unconscious Perceptuo-motor and sensori-actuator processing.[5]
- The levels of our architecture are semi-autonomous and processed in parallel.[6]
- 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 sensory perceived objects, properties, events, and states of affairs in the world and motor capabilities on the other hand. We call this correspondence *alignment*.
- Our architecture may be appropriate both for modeling elephants and for modeling chess-playing agents.

2 The GLAIR Architecture

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

2.1 Related Work

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

2.1.1 Theoretical Issues

We believe that behavior-based AI has adopted the right treatment of every day 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. We 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 (MHP) is a cognitive model [CMN83] that suggests the three components of *perception, cognition*, and *motor*. Cognition consists of working memory, long-term memory, and the cognitive processor. Perception is a hierarchy of sensory processing. Motor 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. Despite this, systems

[5] We 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, we 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). We reserve the term *sub-conscious* for implicit cognitive processes such as category subsumption in KRR systems. See [Sha90] for a discussion of sub-conscious reasoning.

[6] This autonomy is similar to Brooks's subsumption architecture [Bro85], but at a more macroscopic level. Brooks does not distinguish between the three levels we describe, as his work is solely concerned with behaviors whose controlling mechanism we would situate at the Perceptuo-Motor level.

like SOAR adhere to this model. In our architecture, we purposively 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 to selectively inhibit lower-level ones as well, but we have found no good reason to do so yet.

A situated agent, at any moment, attends to only 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 [AC87] proposed indexical-functional representations (which [Agr88] 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 relationships. Although the objects in the environment come and go, the representations of entities and relationships remains the same. For example, the-cup-that-I-am-holding[7] 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 KRR abilities can and should generate deictic representations to create and maintain a focus on entities in the world, but we have not yet designed an implementation strategy.

2.1.2 Implemented Architectures

Brooks's *subsumption* architecture, [Bro85, Bro87, Bro90], clusters behaviors into layers. 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. 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 non-conscious spheres. In anthropomorphic terms, Brooks's agents are all non-conscious. 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 [Mae91]. 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.

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 [Agr88], 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. An (unnamed) behavior module can be thought of as a set of tuples (P,A) where P is a set of ground sensations and A is an instance of an act. For instance, reaching for an object might be a set of tuples (vision/sonar data, wheel motor actuation). After learning, this new behavior module will become active only if the ground sensations match any of the ground 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. We will return to this point briefly in section 3.3.1.

The Servo, Subsumption, Symbolic (SSS) architecture [Con92] is a hybrid architecture for mobile robots

[7] 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.

that integrates the three independent layers of servo control, 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 (i.e., the rate at which it is in real-time contact with the world) with the outside 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 seems to be a decision table versus a general KRR as intended in GLAIR. Unlike GLAIR, SSS assigns particular tasks for each layer and uses a hard-wired interconnection channel among layers.

Albus et al's hierarchical control architecture [ABN81] is 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. By descending down the hierarchy, tasks are decomposed into robot-motion primitives. This differs from our architecture, which is not strictly top-down controlled. 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 differences. The differences are used for planning at higher levels.

Payton in [Pay86] 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 $\langle virtualsensor, reflexivebehavior\rangle$. 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 [RP89].

Rosenschein and Kaelbling's work [Kae88, KR90] describes tools (REX, GAPP, RULER) that, given task descriptions of the world, 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. We should note that unlike Brooks's behavior modules, situated automata use internal states, so their decisions are not Markovian (i.e., they are not ahistoric). They are mainly intended to produce circuits that operate in realtime, and some properties of their operation are provable. The mechanism for generating situated automata, although impressive, seems to inflexible for autonomous agents that have to operate in a wide variety of (possibly unknown) circumstances . Perhaps the operation of our Perceptuo-motor level could be modeled by a situated automaton, but we are not convinced that this is the right formalism to use, due to its inflexibility.

Gat in [Gat91] 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 [Fir87]. The deliberation layer is the least described layer. As for situated automata, we are not convinced that this is the right kind of formalism to use, for the same reasons.

An architecture for low-level and high-level reactivity is suggested in [Hex89]. 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.

SOAR [LNR87] 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

[LHYT91]. In this framework, planning and acting is uniformly represented and controlled in SOAR. This approach lacks the ability of our architecture for generating behavior at non-conscious levels as well as the conscious level (or at different levels in general), and for having different-level behaviors interact in an asynchronous fashion. It also lacks our multi-level representations.

Simmons's Task Control Architecture (TCA) [Sim90] 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. This type of "stepwise refinement" follows effortlessly from our architecture, without the need to explicitly implement it. Since conscious planning which goes on at the Knowledge level uses a more coarse-grained world model, there is simply no possibility to express fine details of planning and execution. These can only be represented and/or computed at the lower Perceptuo-Motor level and Sensori-Actuator level. Planning and execution in our architecture may proceed in a lock-step fashion, but they need not be. TCA uses a message-passing scheme among modules that allows concurrent execution of tasks. It has been used to control the six-legged walking robot Ambler and a cup-collecting robot.

2.2 Architecture Levels

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

2.2.1 Motivation

The three levels of our architecture are of organizational as well as theoretical importance. Organizationally, 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 former. Routine, 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, we have 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 incapacitatedness. Lower levels can function to some extent without higher-level control, and higher levels can function to some extent without lower-level input.[8]

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".[9]

[8] 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.

[9] 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 [Pol92].

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 [Har90, Lak87, Suc88] of actions, concepts, and reasoning is mainly the Perceptuo-Motor level and to some extent the Sensori-Actuator level.

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

2.2.2 The Knowledge Level

The *Knowledge level* 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 this 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 not required. Only if the detail becomes important to 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 this 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 our use of the term "world model" is that we do not intend to have a precise model of all objects in the environment. Instead, we 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.[10] 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.

2.2.3 The Perceptuo-Motor Level

The *Perceptuo-Motor level* uses a more fine-grained representation of events, objects, and states of affairs. For instance, they specify 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. 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 categorial 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 [Har90].

The representations at this level are *embodied* (cf. [Lak87]), meaning that they depend on the body of the agent, its particular dimensions and characteristics. Robots will therefore have different representations

[10] See [SR87] for our use of "intensional representation".

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 in terms of how much torque to apply to an object to lift it,[11] 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.

At the Perceptuo-Motor level, an agent has a close coupling between its behaviors, i.e., responses, and stimuli, i.e., significant world states. 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. We discuss an implementation mechanism for these automated behaviors later in this paper.

2.2.4 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

[11] 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 we 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.

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 they take precedence over any other behavior. 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).[12][13] 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.

2.3 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 [McD91, 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. As noted above, 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* [Har90]. 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 [Har92]. 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 categorial perception (and perhaps "categorial 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 which requires the referents of symbols to be in the world; rather, they are system-internal, similar to what Hausser proposes [Hau89], or what Harnad calls iconic representations: "proximal sensory projections of distal objects, events, and states of affairs in the world" [Har90]. 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 2.4), 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.[14] 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, grosso modo at least (cf. [Rap88]). 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

[12] 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.

[13] 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.

[14] It is never a problem as long as agents need not communicate with the outside world (other agents), of course, cf. [Win75].

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 same.[15] 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.

2.4 Embodied Representation

In section 2.2.3 we already mentioned the use of embodied representations at the Perceptuo-Motor level. We now look at the principle of embodiment from a more abstract point of view.

One of the high-level motivations behind our work is the desire to be able to "program" a robotic autonomous agent by requesting it to do something at 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.[16] 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 embodied concepts, 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").

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 [LHS93].

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

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

2.5 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 to explicitly represent the existence of the object in the Knowledge level while processing sensory information about the object in the Perceptuo-Motor level. Similar to 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 [SR87].

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 PM-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. We believe that 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 [CH93].

2.6 Consciousness

As we 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, 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.[17] 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 (section 3).

Knowledge in GLAIR can migrate from conscious to unconscious levels. In [HCBS93] 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 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.[18] The implementation mechanisms we use for the unconscious levels, such as PMAs, are much faster and require much less 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 (section 3.1).

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

[18] 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. [Lev88]). Elephants don't stand a chance.

3 Applications

Our architecture as described in section 2 can be populated with components that make up the machinery for mapping sensory inputs to *response* actions, as does Russell in [Rus91]. We now discuss some applications of GLAIR that we are currently developing.

Some important general features of GLAIR-agent are the following:

- Varieties of behaviors are integrated: We distinguish between deliberative, reactive, and reflexive behaviors. At the unconscious level, behavior is generated by mechanisms with the computational power of a finite state machine (or less), whereas, at the conscious level, behavior is generated via reasoning (of Turing Machine capabilities). As we move down the architectural levels, computational and representational power (and generality) is traded off for better response time and simplicity of control. Embodied representations aid in this respect (section 2.4).

- We assume agents to possess a set of primitive motor capabilities. The motor capabilities are primitive in the sense that (a) they cannot be further decomposed, (b) they are described in terms of the agent, and (c) no reference is made to external objects. The second property of motor capabilities is so that the success of performing an action should depend only on the agent's bodily functions and proprioceptive sensing. For example, for a robot arm, we might have the following as its motor abilities: calibrate, close-hand, raise-hand, lower-hand, move.

- Our architecture provides a natural framework for modeling four distinct types of behavior, which we call reflexive, reactive, situated, and deliberative. Reflexive and reactive behaviors are predominantly unconscious behaviors, whereas situated and deliberative actions are conscious behaviors.

Reflexive behavior[19] occurs when sensed data produces a response, with little or no processing of the data. A reflex is immediate. The agent has no expectations about the outcome of its reflex. The reflexive response is not generated based on a history of prior events or projections of changing events, e.g., a gradual temperature rise. Instead, reflexive responses are generated based on spontaneous changes in the environment of the agent, e.g. a sudden sharp rise in temperature. In anthropomorphic terms, this is innate behavior that serves directly to protect the organism from damage in situations where there is no time for conscious thought and decision making, e.g., the withdrawal reflex when inadvertently putting one's hand into a fire. Reflexive behavior does not require conscious reasoning or detailed sensory processing, so our lowest level, the Sensori-Actuator level, is charged with producing these behaviors. Our initial mechanism for modeling reflexive behavior is to design processes of the form $T \mapsto A$, where T is a trigger and A is an action. A trigger can be a simple temporal-thresholding gate. The action A is limited to what can be expressed at the Sensori-Actuator level, and is simple and fast.

Reactive behavior requires some processing of data and results in *situated action* [Suc88]. However, its generation is subconscious. *Situated action* refers to an action that is appropriate in the environment of the agent. In anthropomorphic terms, this is learned behavior. An example would be gripping harder when one feels an object is slipping from one's fingers, or driving a car and tracking the road. We use the term *tracking* to refer to an action that requires continual adjustments, like steering while driving. Examples of this type of reactive behavior are given in [Pay86, AHC91]. Situated behavior requires assessment of the state the system finds itself in (in some state space) and acting on the basis of that. It might be modeled by the workings of a finite state automaton, for example, the Micronesian behavior described in [Suc88]. Situated action is used in *reactive planning*[AC87, Fir87, Sch87].

We have developed an implementation mechanism for the PM-level which we call PM-automata (PMA), [HN92]. A PMA is a finite state machine in which each state is associated with an act and arcs are associated with perceptions. In each PMA, a distinguished state is used to correspond to the no-op act. Each state also contains an auxiliary part we call Internal State (IS). An IS is used in arbitrating among competing arcs. Arcs in a PMA are situations that the agent perceives in the environment. When a PMA arc emanating from a state becomes active, it behaves like an asynchronous interrupt to the act in execution in the state. This causes the PMA to stop executing the act in the state and to start executing the act at the next state

[19] E.g., visual reflexes in [RB78]: Here responses are generated to certain visual stimuli that do not require detailed spatial analysis.

at the end of the arc connecting the two states. This means that in our model the agent is never idle, and it is always executing an act. The primary mode of acquiring PMAs in GLAIR is by converting plans in the Knowledge level into PMAs through a process described in [HN92]. A PMA may become active as the result of an intention to execute an action at the Knowledge level. Once a PMA becomes active, sensory perception will be used by the PMA to move along the arcs. The sensory perceptions that form the situations on the arcs as well as subsequent actions on the PMA may be noticed at the Knowledge level. In general, the sensory information is filtered into separate streams for PMAs and for the Knowledge level.

3.1 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 2 represents the setup.

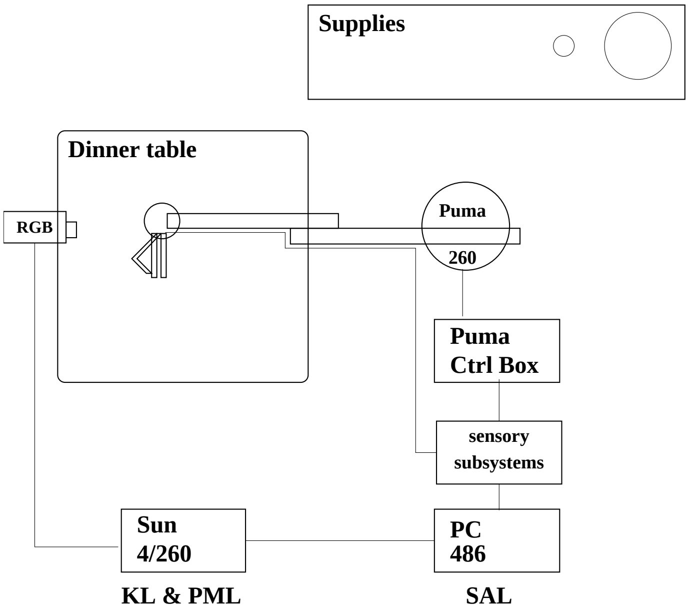

Figure 2: The Robot Waiter physical setup.

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 3 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.

Actions are transmitted from KL to PML, e.g., *look for a red apple* and *pick it up*. Once the sensory memory contains an object matching the object desired, actions involving that object can be understood at

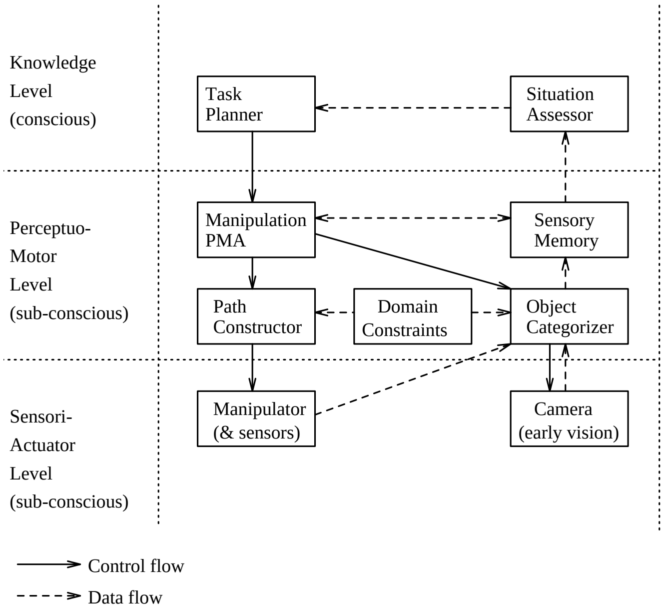

Figure 3: Schematic representation of the Robot Waiter GLAIR-agent.

the PML. For instance, once a red apple 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 apple, grasp it, and lift it. A PMA is a implementation mechanism for routine activities at an "unconscious" level, [HN92, HLS92, HCBS93]. 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 [BK69] 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 [Lam]. 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.

The main components of the Robot Waiter system are listed below:

- A World model: In the general KRR system, we maintain a conscious world model.[20] This model explicitly represents the agent's knowledge (or beliefs) about the entities in its world. Representations at this level may or may not correspond to representations at the Perceptuo-Motor level, as explained in section 2.2. As much as it might be desirable to avoid building internal models of the world,[21] having some modeling capacity is necessary, we believe.

- Specialized knowledge bases: for instance knowledge about action selection, planning, learning, experimentation, and perception.

- A Kinematic/Perceptual model: At the Perceptuo-Motor level, we maintain a model of the simple agent-level physics of the objects of interest to the agent.[22] The kinematic/perceptual model models

[20] Albus defines a world model as the agent's best estimate of objective reality [Alb91].
[21] Situated cognition and reactive planning are proponents of avoiding world modeling, e.g., [Bro90, Suc88].
[22] Even the prominent advocators of doing away with world models actually use a variety of models, some of which qualify as

motor capacities and motor memory that might be implemented in different ways (e.g., purely procedurally or in a network of nodes and weighted links), but we prefer the declarative approach for its ease of interpretation and debugging. It also contains perceptual representations of perceived objects. Representations at this level are embodied and agent-centered (section 2.4).

- Reactive processes: Also at the Perceptuo-Motor level, a number of independent processes monitor perceptual inputs, and control reactive behaviors of the agent. We need a mechanism for arbitrating among various reactive and other processes at the same level, which we have not worked out yet.

- Primitive motor and sensory actions: At the Sensori-Actuator level, a number of these primitive actions are implemented. A primitive motor action might be "move ahead", and a primitive sensory action might be "look at position $\langle x, y, z \rangle$ (which in turn may involve primitive motor actions). Sensing as such is not considered a primitive sensory *action*, and it goes on continuously.

- Reflexes: Also at the Sensori-Actuator level, a number of reflexes are implemented as low-level independent processes that monitor raw sensory data and can control actuators directly, temporarily pre-empting higher level control.

As of this writing, the Robot Waiter project is partially implemented, but not operational yet.

3.2 A simulation study: Air Battle

We are interested in modeling behavior generation by agents that function in dynamic environments. We make the following assumptions for the agent:

- The environment demands continual and rapid acting, e.g., playing a video-game.
- The impact of the agent's actions depends on the situations under which actions are applied and on other agents' actions.
- Other agents' actions are nondeterministic.
- The agent does not know about long term consequences (i.e., beyond the current situation) of its actions.
- The agent is computationally resource bound. We assume that the agent needs time to think about the best action and in general there is not enough time.

To cope in dynamic environments, an agent which is resource bound needs to rely on different types of behaviors, for instance, reflexive, reactive, situated, and deliberative behaviors. Reflexive and reactive behaviors are predominantly "unconscious" behaviors, situated action may be either "unconscious" or "conscious", and deliberative actions are predominantly "conscious" behaviors. We assume that in general "conscious" behavior generation takes more time than "unconscious" behavior generation.

We have written a program, Air Battle Simulation (ABS), that simulates world war I style airplane dogfights. ABS is an interactive video-game where a human player plays against a computer driven agent. The game runs on SparcStations and starts up by displaying a game window and a control panel window (figure 4). 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

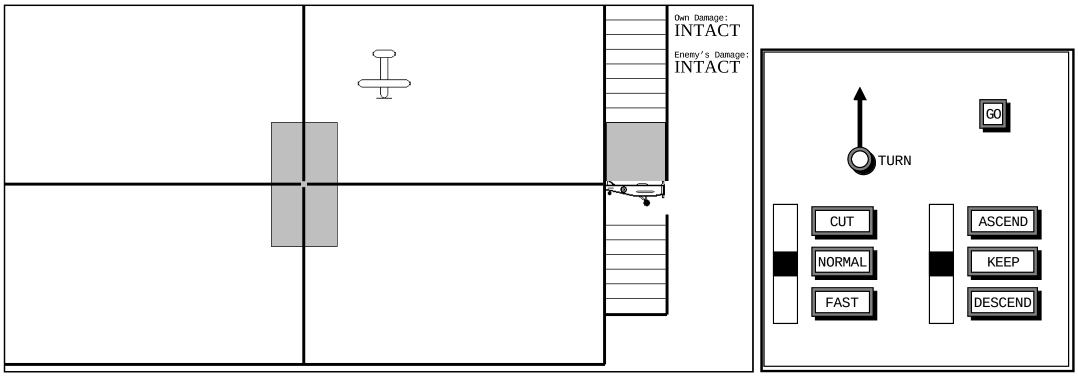

Figure 4: Air Battle Simulation game window and control panel (see text).

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.

The agent is developed in accordance with the principles of the GLAIR architecture. Figure 5 schematically represents its structure. Initially, the agent has not acquired a PMA, and uses conscious level reasoning

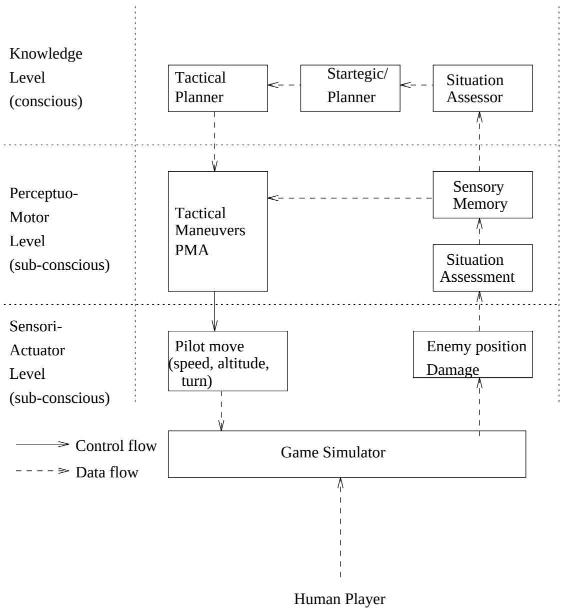

Figure 5: Schematic representation of the Air Battle Simulation GLAIR-agent.

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 can be 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 are exploring these learning issues. 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

kinematic/perceptual models. For instance, see Chapman's work on Sonja [Cha90] where Sonja has to build a convex hull of obstacles and compute angles in order to decide the best way to avoid them.

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.

3.2.1 Improving "Unconscious" Behaviors

The rules of a PMA are pairs of situation/action. As it turns out, a situation can be paired up with multiple actions. The object of learning here is to learn which actions when associated with a situation yield a better result, i.e., the pilot ends up in a more desirable situation.

Some situations in ABS are more desirable for the pilot than others, e.g., being right behind the enemy and in shooting range. Let's assume that we can assign a goodness value to each situation s between -1 and 1, **G**(s). As the pilot makes a move, it finds itself in a new situation. This new situation is not known to the pilot since it also depends on the other pilot's move. Since the new situation is not uniquely determined by the pilot's move, the pilot's view of the game is not markovian.

Q(s,a) is the evaluation of how appropriate action a is in situation s. **R**(s,a) is the goodness value of the state that the pilot finds itself after performing a in situation s. **R**(s,a) is determined as the game is played and cannot determined beforehand. This is called the immediate reward. γ is a parameter between 0 and 1 that we plan to vary that to determine how important it is to be in the state that the pilot ends up in after his move. In reinforcement based learning this is known as the discount factor. we let $\mathbf{Q}$(s,a) = $\mathbf{R}$(s,a) + γ $\max_k \mathbf{Q}$(s',k) where situation s' results after the pilot performs a in s. At the start of game, all **Q**(s,a) in the PMA are set to 1. As the game is played, **Q** is updated. As of this writing we are experimenting with setting appropriate parameters for **Q**.

3.2.2 Observing Successful Patterns of Interaction in the World

We assumed that the agent does not know about long term consequences of its actions. Furthermore, the reinforcement based learning we described in the previous section assumes a markovian environment. That is, the agent believes the world changes only due to its own actions. This makes it necessary to observe interactions with the world in order to learn sequences of 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 reoccur, 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, [Hex92].

We plan to explore other learning techniques such as experimentation as a form of learning [She89]. We are also interested in developing experiments that will help in psychological validation of GLAIR and the learning strategies used in ABS. As of the time of writing ABS is fully operational, but several issues are still being investigated, as noted above.

3.3 A simulation study: the Mobile Robot Lab

We now describe the Mobile Robot Lab (MRL), a simulation environment we are developing for mobile robots that function as GLAIR-conformant autonomous agents. 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. To make the simulation more realistic and less predictable, some stochastic properties are built in. More than one agent can be accommodated. 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).

3.3.1 Emergent Behaviors

A major objective for this project is learning emergent behaviors. Like Agre with his improvised actions [AC87] and Brooks with his subsumption architecture [Bro85] we 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. Furthermore, it did not attempt to learn the improvised behavior. In this simulation, we plan to start with a minimal number of primitive actions and sensations. Our basis for this

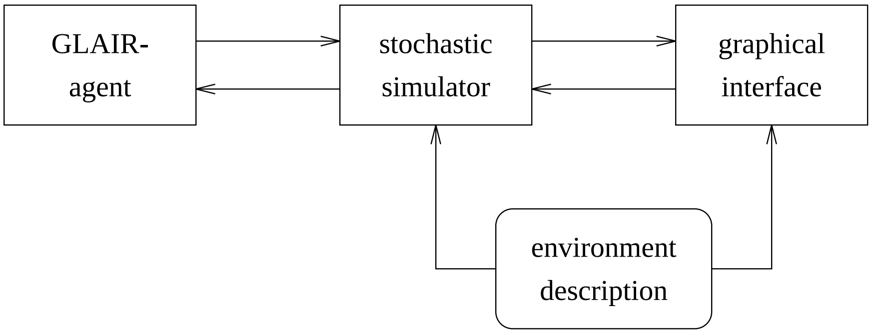

Figure 6: Overview of a complete setup using MLR. It consists of a GLAIR-agent, a simulator with an incorporated model of a physical environment, and a graphical interface. Arrows represent direction of data flow among the components.

minimality and the choice of primitive actions is physiological. In other words, in our modeling an agent, we will choose actions that are physically basic for the agent's body as primitive. We then instruct the agent to perform tasks and in the midst of accomplishing this, we expect it to notice some types of behaviors emerge. An example of an emergent behavior we will explore is *moving toward an object*. We expect the agent to be learning to coordinate its wheel motions, starting from nothing more than the primitive sensation of contact with an external object, and the primitive actions of turning its motors independently on or off.

3.3.2 The physical environment description

The simulator uses a description of the physical environment that the simulated robot operates in. This description is easily modifiable (without reprogramming). It includes the physical characteristics of the mobile robot and the space in which it moves. A 2D bird's eye view of a typical room setup with a robot inside is show in figure 7.

The room the robot moves is 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.

3.3.3 The simulator

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. It also prevents the robot from going "through" walls or objects. It 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 simulator incorporates a simplified lighting model to determine the appearance (color) of objects in the room. Light sources can either be point sources or homogeneous diffuse sources. Each light source has its own SPD. Each object has its own SRF. All objects are assumed to be Lambertian reflectors.

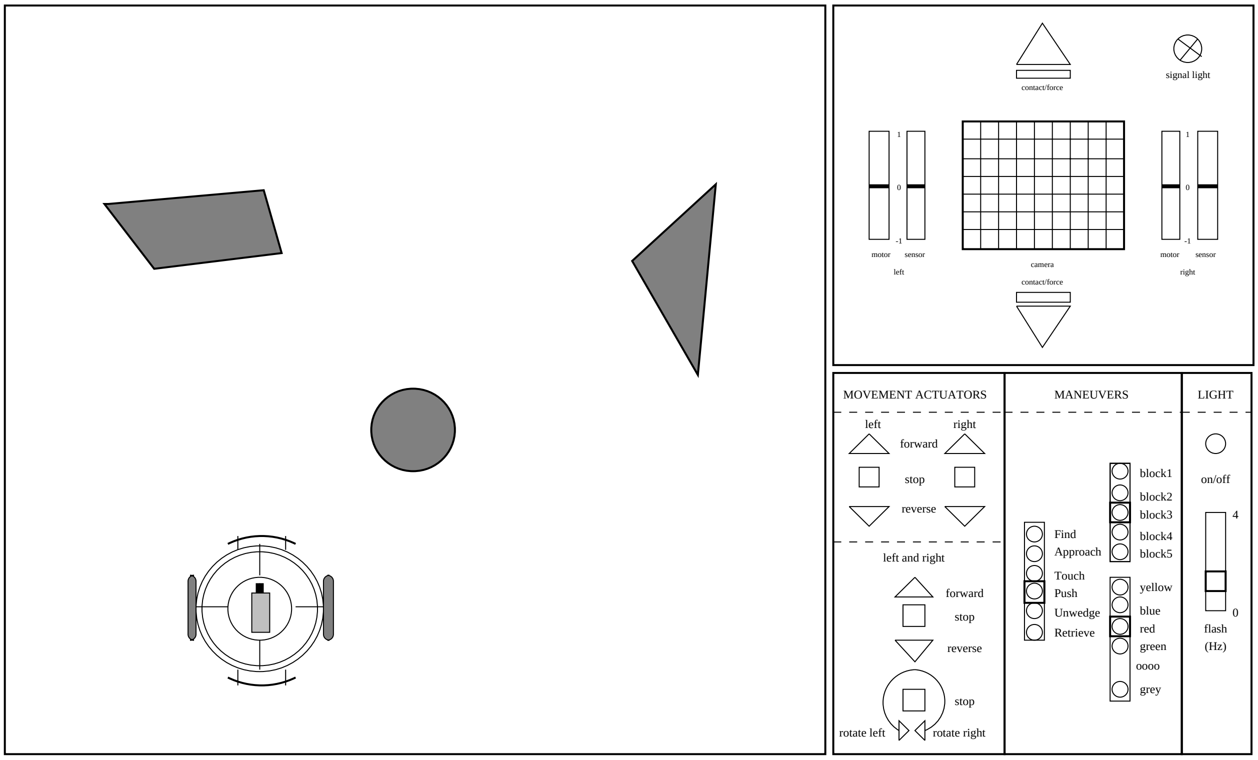

Figure 7: A 2D bird's eye view of a typical room setup with a mobile robot inside (left), the Sensors and Actuators Display (upper right) and the Movement Control Panel (lower right). See text.

To enhance the realism of the simulation and to introduce some uncertainty into the environment, all robot controls and sensors work stochastically. Sensor readings vary stochastically over time around the "true" value, and the same is true for actuator output. We assume normal distributions for all such variations, with variable standard deviations.

3.3.4 The graphical interface

The graphical interface provides a real-time view of the robot and the robot's environment, using data obtained from the simulator (figure 7). 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 (not shown in figure 7). The graphical interface also contains a control panel display which provides manual control over the robot's movements and maneuvers. A movement command for a robot is composed by the user by selecting power levels and directions of rotation for each of the two wheels. A maneuver is a higher-level instruction for the robot. Initially, it can be one of 1) find a block, 2) approach a block, 3) touch a block, 4) push a block, 5) unwedge a block, and 6) retrieve a block. Blocks can be referenced by color (not necessarily unique) or by a numeric identifier. The control panel also allows the robot's signal light to be switched on or off.

3.3.5 The agent

The autonomous agent for this project, with its simulated mobile robot body, conforms to the GLAIR architecture.

Primitive abilities. The agent has the following primitive sensations:

- Speed and direction of rotation for each of the left and right wheel motors are independently sensed. Sensed values for each motor range from 2 foot/sec of wheel circumference to -2 foot/sec of wheel circumference (in increments of 0.1).
- The bumpers in the front and the back can sense initial contact between the bumper and external object as well as reaction forces due to pushing. Sensor values range from 0 to 2, in increments of 1.
- The camera inputs an 9x7 array of RGB triplets, each triplet representing an average value over the corresponding portion of the camera's field of view.

The agent has the following primitive actions:

- Speed and direction of rotation for each of the left and right wheel motors are independently controlled. 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 which can bring it to stop. We assume negligible acceleration/deceleration times.
- The light on top of each robot is either on, off, or flashing with a frequency of 1-4 Hz (in increments of 1 Hz).

The Sensori-Actuator level. Sensing and acting are paired up at this level to form reflex behaviors. The following reflex behaviors are implemented. Reflexes have direct access to sensors and actuators, and each may use its own representations and implementation strategies; we list the input and output domains.

- Stop reflex 1: **bumpers** $\mapsto$ **motor controls**; if the robot bumps into anything, it stops moving immediately.
- Force limit reflex: **motor controls** $\times$ **motor sensors** $\mapsto$ **motor controls**; if the sensed speed is significantly different from the speed control settings, the robot stops moving. This may occur when something is holding the robot back, for instance.
- Attention reflex: **camera input** $\mapsto$ **motor controls**; if anything moves in the robot's peripheral visual field, it stops and turns towards the location of movement.
- Stop reflex 2: **camera input** $\mapsto$ **motor controls**; if anything comes close in the robot's field of view, it stops moving.
- Escape reflex: **camera input** $\mapsto$ **motor controls**; if anything approaches the robot fast, the robot runs away.[23]

The Perceptuo-Motor Level. At the Perceptuo-Motor level, primitive sensations and actions are processed and combined to form more complex perceptions and behaviors. We list some of these below. We are currently investigating the use of foveal vision for the agent, which would alter some of the descriptions below.

A signal, object-seen, is generated to denote having seen an object. Signals will be generated to denote that the object is in the right, center, or left field of view. If an object in the field of view becomes larger (due to getting closer), a signal is generated that the object is object-bigger. Similarly, the object in the field of view will get smaller when departing from the object, signaled by object-smaller. If the object becomes an obstacle for the robot, the vision system generates an object-too-close signal. If anything moves in the robot's peripheral visual field, it will generate the signal moving-object (together with a position signal). Another complex perception is the trajectory of robot movement. As the robot senses its wheel speed and direction of rotation combined with visual input, it builds a trajectory of movement. This can be used in building a map of the domain.

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

[23] Inhibiting this reflex might turn the agent from a chicken with a long life expectancy into an eagle with short life expectancy.

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 as a PMA. For instance, touching an object will evolve into a PMA. Even after the initial learning of a PMA for a complex action, learning will continue to improve the PMA.

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 is a list of behaviors and percepts we want the robot to learn. We will provide the robot with appropriate rewards for these behaviors. Below we give a list of emergent behaviors that the robot will learn completely on its own (no rewards).

- Touch behavior: approach an object until contact is made.
- Proximity percepts: proximity to an object, derived from camera data. (e.g., see all the signals we listed above in the example of touching behavior).
- 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

When more than one agent is present, behaviors at this level may include finding or hiding from other agents, following or running from other agents, playing games (hide and seek, for instance).

Earlier we defined behaviors in terms of PMAs. we consider subsets[24] of PMAs to also be behaviors. Percepts and behaviors listed below are emergent, i.e., they are learned but the agent did not intend to learn them. The robot is never told about these behaviors. They are learned in PMAs, as in the PMA for touching objects.

- Proprioceptive movement percepts and movement behaviors: moving forward, backward, turning left forward, turning right forward, turning left backward, turning right backward, rotating left, and rotating right. Each of these is a percept as well as a behavior.

Some other percepts and behaviors situated at the PM level are listed below. They are hard-wired in the initial implementation, but could conceivably be learned or emergent as well. As a rule of thumb, we consider the perceptual representation required to perform discrimination tasks to be hard-wired, and those required to perform identification tasks to be learned (or learnable).

- Color percepts: in perceptual color space coordinates, derived from camera data.
- Color hallucination behavior: imagine the surroundings in different colors by warping the perceptual color space.

[24] Let's define a subset of a PMA to be a disjoint PMA, i.e., B is a subset of A iff all components of B are subsets of components of A and no transitions exist between actions in B and thoses in A-B.

- Shape percepts: simple object shapes like square, rectangle, tall, short, narrow, wide, irregular, big, small, derived from camera data.

Also at this level, basic emotions like fear and curiosity may be implemented as particular types of behaviors (aligned with symbolic labels at the Knowledge level).

The Knowledge Level. 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.

4 Concluding Remarks

We have presented a general architecture for autonomous agents that integrates behavior-based architectures with traditional architectures for symbolic systems. The architecture specifies how an agent establishes and maintains a conscious connection with its environment while mostly unconsciously processing sensory data, and filtering information for conscious processing as well as for reflexive and reactive acting. We ended our paper by instantiating the architecture with several (physical and simulated) agents embedded in their environment, in various stages of implementation. We believe our work can contribute towards integrating traditional ungrounded symbol systems with the newer physically grounded systems. Combining an elephant's body with a man's[25] mind makes for an awesome combination.

5 Acknowledgements

We appreciate comments made on an earlier draft of this paper by John Pollock, Beth Preston, Donald Nute, William Rapaport, Stevan Harnad, Chris Brown, Phil Agre, and Tim Smithers. Goofs are entirely attributable to the authors, of course.

References

[ABN81] James Albus, Anthony Barbera, and Roger Nagel. Theory and practice of hierarchical control. In *23rd International IEEE Computer Society Conference*, pages 18–38, 1981.

[AC87] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of AAAI-87, Seattle, Wa.*, pages 268–272, July 1987.

[Agr88] Philip Agre. The dynamic structure of everyday life. Technical Report 1085, MIT Artificial Intelligence Laboratory, MIT, 1988.

[AHC91] Scott Anderson, David Hart, and Paul Cohen. Two ways to act. In *ACM SIGART Bulletin*, pages 20–24. ACM publications, 1991.

[Alb91] James Albus. Outline for a theory of intelligence. In *IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, No. 3*, pages 473–509, 1991.

[And83] J. R. Anderson. *The Architecture of Cognition.* Cambridge: Harvard University Press, 1983.

[BB82] Dana H. Ballard and Christopher M. Brown. *Computer Vision.* Prentice-Hall, Englewood Cliffs, NJ, 1982.

[25] He-man or She-man.

[BK69] Brent Berlin and Paul Kay. *Basic Color Terms: Their Universality and Evolution.* University of California Press, Berkeley CA, first paperback edition, 1991 (orig. 1969).

[Bro85] Rodney Brooks. A robust layered control system for a mobile robot. Technical Report 864, MIT AI Labs, MIT, 1985.

[Bro87] Rodney Brooks. Planning is just a way of avoiding figuring out what to do next. Technical Report 303, MIT AI Labs, 1987.

[Bro90] Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[CH93] Guido Caicedo and Henry Hexmoor. Alignment in GLAIR. Technical Report Forthcoming, Computer Science Department, State University of New York at Buffalo, Buffalo, NY, 1993.

[Cha90] David Chapman. Vision, instruction, and action. Technical Report Technical Report 1204, MIT Artificial Intelligence Laboratory, MIT, 1990.

[CMN83] S.K. Card, T.P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction.* Erlbaum, Hillsdale, N.J., 1983.

[Con92] Jonathan Connell. Sss: A hybrid architecture applied to robot navigation. In *IEEE Conference on Robotics and Automation*, pages 2719–2724, 1992.

[Cul63] James Culbertson. *The Minds of Robots.* U. of Illinois Press, 1963.

[Fir87] R. James Firby. An investigation into reactive planning in complex domains. In *Proceedings of AAAI-87*, pages 202–206, 1987.

[Fod83] Jerry Fodor. *The Modularity of Mind.* MIT Press, 1983.

[Gat91] Erann Gat. Reliable goal-directed reactive control of autonomous mobile robot. Technical Report Tec, Dept. of Computer Science, Virginia Polytechnic Institute and State University, 1991.

[Har90] Stevan Harnad. The symbol grounding problem. *Physica D*, 42(1-3):335–346, 1990.

[Har92] Stevan Harnad. Electronic symposium on computation, cognition and the symbol grounding problem. E-mail symposium (ftp archive at princeton.edu:/pub/harnad/sg.comp.arch*), 1992.

[Hau89] Roland Hausser. *Computation of Language: An Essay on Syntax, Semantics and Pragmatics in Natural Man-Machine Communication.* Springer-Verlag, New York, NY, 1989.

[HCBS93] Henry Hexmoor, Guido Caicedo, Frank Bidwell, and Stuart Shapiro. Air battle simulation: An agent with conscious and unconscious layers. In *UBGCCS-93.* Dept. of Computer Science, SUNY at Buffalo, New York, 1993.

[Hex89] Henry Hexmoor. An architecture for reactive sensor-based robot. In *NASA Goddard conference on AI*, Greenbelt, MD, 1989.

[Hex92] Henry Hexmoor. Representing and learning successful routine activities. Technical Report Unpublished PhD Proposal, Dept. of Computer Science, SUNY at Buffalo, New York, 1992.

[HLS92] Henry Hexmoor, Joe Lammens, and Stuart Shapiro. 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, 1992.

[HN92] Henry Hexmoor and Donald Nute. Methods for deciding *what to do next* and learning. Technical Report AI-1992-01, AI Programs, The University of Georgia, Athens, Georgia, 1992. Also available from SUNY at Buffalo, CS Department TR-92-23.

[Kae88] Leslie Kaelbling. Goals as parallel program specifications. In *Proceedings of AAAI-88.* Morgan Kaufman, 1988.

[KR90] Leslie Kaelbling and Stanley Rosenschein. Action and planning in embedded agents. In Pattie Maes, editor, *Designing Autonomous Agents*, pages 35–48. MIT Press, 1990.

[Lak87] George Lakoff. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind.* University of Chicago Press, Chicago, IL, 1987.

[Lam] Johan M. Lammens. 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.

[Lev88] Hector J. Levesque. Logic and the complexity of reasoning. *Journal of Philosophical Logic*, 17:355–389, 1988.

[LHS93] Johan Lammens, Henry Hexmoor, and Stuart Shapiro. Embodiment in glair: a grounded layered architecture with integrated reasoning for autonomous agents. In *Proceedings of the Florida AI Research Symposium 93*, page to appear, 1993.

[LHYT91] J. Laird, M. Huka, E. Yager, and C. Tucker. Robo-soar: An integration of external interaction, planning, and learning, using soar. In *Robotics and Autonomous Systems*, 1991.

[LMA91] Pat Langley, Kathleen McKusick, and John Allen. A design for the icarus architecture. In *ACM SIGART Bulletin*, pages 104–109. ACM publications, 1991.

[LNR87] J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.

[Mae91] Pattie Maes. Action selection. In *Proceedings of Cognitive Science Society Conference*, 1991.

[McD91] Drew McDermott. Robot planning. Technical Report CS-861, Yale University, 1991.

[MRH86] J. L. McClelland, D. E. Rumelhart, and G. E. Hinton. The appeal of parallel distributed processing. In David Rumelhart, James McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing*, chapter 1, pages 3–44. MIT Press, Cambridge MA, 1986.

[Pay86] David Payton. An architecture for reflexive autonomous vehicle control. In *Proceedings of Robotics Automation*, pages 1838–1845. IEEE, 1986.

[Pol89] John Pollock. *How to Build a Person.* MIT Press, 1989.

[Pol92] John Pollock. New foundations for practical reasoning. In *Minds and Machines*, 1992.

[Rap88] William J. Rapaport. Syntactic semantics: Foundations of computational natural-language understanding. In James H. Fetzer, editor, *Aspects of Artificial Intelligence*, pages 81–131. Kluwer Academic, New York NY, 1988.

[RB78] D. Regan and K.I. Beverly. Looming detectors in the human visual pathways. In *Vision Research 18*, pages 209–212. 1978.

[RP89] J. Kenneth Rosenblatt and David Payton. A fine-grained alternative to the subsumption architecture for mobile robot control. In *Proceedings of Internations Joint Conference on Neural Networks*, 1989.

[Rus91] Stuart Russell. An architecture for bounded rationality. In *ACM SIGART Bulletin*, pages 146–150. ACM publications, 1991.

[Sch87] Marcel J. Schoppers. Universal plans for unpredictable environments. In *Proceedings 10th IJCAI*, pages 1039–1046, 1987.

[Sha90] Stuart C. Shapiro. Cables, paths, and 'subconscious' reasoning in propositional semantic networks. In *Principles of Semantic Networks.* Morgan Kaufman, 1990.

[She89] Wei-Min Shen. *Learning from the Environment Based on Actions and Percepts.* PhD thesis, Carnegie Mellon University, 1989.

[Sim90] Reid Simmons. An architecture for coordinating planning, sensing, and action. In *Proceedings the DARPA workshop*, pages 292–297, 1990.

[SR87] S. C. Shapiro and W. J. Rapaport. Sneps considered as a fully intensional propositional semantic network. In N. Cercone and G. McGalla, editors, *The knowledge frontier: essays in the representation of knowledge*, pages 262–315. Springer, New York, 1987.

[Suc88] Lucy A. Suchman. *Plans and Situated Actions: The Problem of Human Machine Communication.* Cambridge University Press, 1988.

[Win75] Patrick Henry Winston. Learning structural descriptions from examples. In Ronald J. Brachman and Hector J. Levesque, editors, *Readings in Knowledge Representation*, pages 141–168. Morgan Kaufmann, San Mateo CA, 1985 (orig. 1975).