Of Elephants and Men

Johan M. Lammens[1], Henry H. Hexmoor[2], and Stuart C. Shapiro[2]

[1] Advanced Robotics Technology and Systems Laboratory, Scuola Superiore S. Anna, 56127 Pisa, Italy; e-mail lammens@arts.sssup.it
[2] Computer Science Department, State University of New York at Buffalo, NY 14260, USA; e-mail {hexmoor|shapiro}@cs.buffalo.edu

Abstract. In the elephant paper, Brooks criticized the ungroundedness of traditional symbol systems and proposed physically grounded systems as an alternative. We want to make a contribution towards integrating the old with the new. We describe the GLAIR 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 present some agent components that we place in our architecture to build agents that exhibit situated activity and learning, and some applications. We believe that the Brooksian behavior generation approach goes a long way towards modeling elephant behavior, which we find most interesting, but that in order to generate more deliberative behavior we need something more.

1 Introduction and Overview

In the elephant paper [11] 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 [11], in particular:

- The ungroundedness of symbolic systems (referred to as "the symbol grounding problem" by [19]): 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 (KRR) 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 (as opposed to simulated) 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[3], 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.[4] 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 [17], or a *connectionist* point of view, e.g. [46], or an *anti-modularity* assumption as in Brooks's subsumption architecture [9]. 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 [13]. Our architecture is characterized by a three-level organization into a Knowledge Level (KL), a Perceptuo-Motor Level (PML), and a Sensory-Actuator Level (SAL). This organization is neither modular, anti-modular, hierarchical, anti-hierarchical, nor connectionist in the conventional sense. It integrates a traditional symbol system with a physically grounded system, i.e., a *behavior-based* architecture. The most important difference 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 [6], rather than a central vs. peripheral metaphor.

Architectures for building physical systems, e.g., robotic architectures [3], tend to address the relationship between a physical entity, (e.g., a robot), sen-

[3] Grounded Layered Architecture with Integrated Reasoning

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

sors, effectors, and tasks to be accomplished. Since these physical systems are performance centered, they often lack general knowledge representation and reasoning mechanisms. 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. One might say these systems are not concerned with *consciousness*. These architectures address what John Pollock calls *Quick and Inflexible* (Q&I) processes [49]. 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.[5]

Architectures for understanding/modeling behaviors of an anthropomorphic agent, e.g., cognitive architectures [4, 49, 42], 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[6] 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.

There are several features that we hope 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.[7]

[5] A machine like a vending machine or an industrial robot has responses, but it is *unconscious*. See [15] for a discussion of independence of consciousness from having a response. Also, intelligent behavior is independent of consciousness in our opinion.

[6] "Embedded agents are computer systems that sense and act on their environment, monitoring complex dynamic conditions and affecting the environment in goal-oriented ways." ([31] page 1).

[7] 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

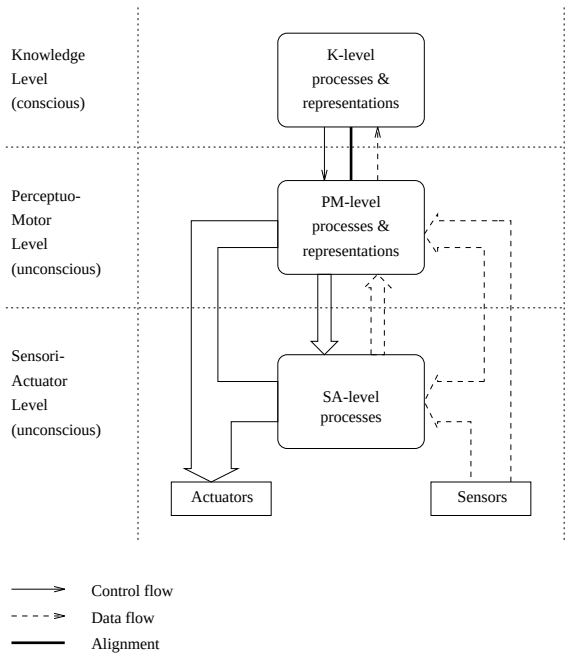


Fig. 1. Schematic representation of the agent architecture. Width of control and data paths suggests the amount of data passing through (bandwidth). Sensors include both world-sensors and proprio-sensors.

- The levels of our architecture are semi-autonomous and processed in parallel.[8]
- 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.

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 [56] for a discussion of sub-conscious reasoning.

[8] This autonomy is similar to Brooks's subsumption architecture [9], 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.

- 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 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 should be appropriate both for modeling elephants and for modeling chess-playing agents.[9]

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.

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 [12] 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 like SOAR adhere to this model. In our architecture, we deliberately avoid this kind

[9] Though not necessarily for chess-playing elephants.

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 [2] proposed indexical-functional representations (which [1] 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[10] 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.

Implemented Architectures. Brooks's *subsumption* architecture, [9, 10, 11], clusters behaviors into layers. Low-level behaviors, like deciding the direction of motion and speed, can be inhibited (subsumed) 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,

[10] 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.

especially if they have to learn from their interactions with the environment. Pattie Maes has experimented with a version of a behavior-based architecture, ANA [44], which 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 [1], 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 grounded 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 grounded sensations match any of the grounded sensations experienced before. As a measure of abstraction and generalization, we may allow near matches for sensations. To bootstrap the learning process, we need a set of primary or first-order ("innate", for the philosophically inclined) sensations and actions. We will return to this point briefly in Sect. 3.3.

The Servo, Subsumption, Symbolic (SSS) architecture [14] is a hybrid architecture for mobile robots 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 system 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 [3] 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 Albus's 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 [48] 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 [53].

Rosenschein and Kaelbling's work [30, 31] 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 real time, and some properties of their operation are provable. The mechanism for generating situated automata, although impressive, seems too inflexible for autonomous agents that have to operate in a wide variety of (possibly unknown) circumstances. Perhaps the operation of 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 [18] 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 [16]. The deliberation layer is the least described layer. As with 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 [22]. 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 [38] 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 [37]. 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) [61] 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. (see the discussion of engaged vs. disengaged reasoning in Sect. 2.6). 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.

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.[11]

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."[12]

In anthropomorphic terms, 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

[11] 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.

[12] 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 [50].

substrate of grounding and embodiment [19, 39, 62] 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.

The Knowledge Level. The *Knowledge level* contains a traditional KRR and/or planning system like SNePS [58, 59], 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 atomic 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 the agent's explicit reasoning 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 non-descript intensional representation [57] of an object. 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.

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 some object identifiers at the Knowledge level and some 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 symbols at the Knowledge level, i.e., sensory-invariant representations constructed from sensory data by the perceptual processor [19].

The representations at this level are *embodied* (cf. [39]), meaning that they depend on the body of the agent, its particular dimensions and characteristics. Robots will therefore have different representations at this level than people would, and different robots will have different representations as well. These representations are agent-centered and agent-specific. For instance, they would not be in terms of grams and meters, but in terms of how much torque to apply to an object to lift it,[13] 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. 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

[13] 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.

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 [33, 34, 36]. 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.

The Sensori-Actuator Level. The *Sensori-Actuator level* is the level of primitive motor and sensory actions, for instance "move from $\langle x, y, z \rangle$ to $\langle x', y', z' \rangle$ or "look at $\langle x, y, z \rangle$". At this level, there are no object representations as there are at the Knowledge level and the Perceptuo-Motor level.

There are no explicit declarative representations of any kind, only procedural representations (on the actuator side) and sensor data (on the sensory side). Primitive motor actions may typically be implemented in a robot control language like VAL, and some elementary data processing routines may be implemented in a sensory sub-system, like dedicated vision hardware. At this level, we also situate *reflexes*, which we consider to be low-level loops from sensors to actuators, controlled by simple thresholding devices, operating independently of higher-level mechanisms, and able to pre-empt the latter. We see reflexes as primitive mechanisms whose main purpose is prevention of damage to the hardware, or to put it in anthropomorphic terms, survival of the organism. As such 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).[14] [15] 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 [47, 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* [19]. 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

[14] 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.

[15] 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.

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 [20]. Grounding is accomplished in our architecture in part through the alignment of the Knowledge and Perceptuo-Motor levels. 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 [21], or what Harnad calls iconic representations: "proximal sensory projections of distal objects, events, and states of affairs in the world" [19]. 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 (Sect. 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.[16] 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. [51]). The problems of translation and communication in general consist at least in part of establishing a correspondence between concepts (and symbols) used by the participants. It is helpful to be able to use referents in the external world as landmarks in the semantic landscape, but one consequence of embodied semantics is that *even* if it is possible to establish these common external referents for symbols, there is still no guarantee that the symbols will actually *mean* exactly the same thing, because in effect the same referent in the world is *not* the same thing to different agents. If we accept this view, it is clear that approaches to semantics based on traditional logical model theory are doomed to fail, because they *presuppose* "identity of referents" and an unambiguous mapping from symbols to referents, the same one for all agents. Another problem is of course the presupposition that all objects are uniquely identifiable. The use of deictic representations does not impose such a condition; as far as our agents are concerned, if it looks and feels the same, it is the same.[17] Nothing hinges on whether or not the objects in the agent's surroundings are *really* ex-

[16] It is never a problem as long as agents need not communicate with the outside world (other agents), of course, cf. [63].

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

tensionally the same as the identical-looking ones that were there a moment ago or will be there a moment later.

In Sect. 3.1 we present an implementation of our architecture that illustrates our ideas on symbol grounding, in the domain of color perception and color naming.

2.4 Embodied Representation

In Sect. 2.2 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 most general motivations behind our work is the desire to be able to "program" a robotic autonomous agent by requesting it to do something and have it "understand", rather than telling it how to do something in terms of primitive motions with little or no "understanding". For instance, we want to tell it to go find a red pen, pick it up, and bring it to us, and not have to program it at a low level to do these things.[18] 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

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

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 in terms of dexterity, 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 [26].

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 at the Perceptuo-Motor level. Similarly, 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 [57].

Having sensed an object, an assertion is made about the object being sensed at the GLAIR Knowledge level. Once the object is no longer in the "field of perception", the assertion about its being sensed is removed. This is tantamount to disconnecting the relationship between the symbolic representation and the world. If at the Perceptuo-Motor level a previously sensed object is again being sensed, we reassert the fact that the object, the same one represented before at 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.

2.6 Engaged and Disengaged Reasoning

Our architecture allows us to elegantly model two different modes of reasoning and planning, which we call *engaged reasoning* and *disengaged reasoning*. Engaged reasoning takes place when all the elementary symbols at the Knowledge level that are involved in the current reasoning activity are immediately aligned with representations at the Perceptuo-Motor level. In practical terms, this means that the agent is reasoning about objects within its field of perception. This may require active perception to keep track of objects, or to shift attention to new objects as the reasoning progresses. For the purpose of object tracking and attention shifting, Knowledge level sensory actions are defined and can be reasoned about like ordinary actions [36]. Engaged reasoning can be done while the actions being reasoned about are actually being carried out, in a kind of plan-as-you-go mode with continuous monitoring of progress being made, or in a more hypothetical mode with no actions being carried out, but potentially affected objects being gauged while the plan is being developed.

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

2.7 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.[19] 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

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

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 [24] 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.[20] 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.

3 Applications

Our architecture as described in Sect. 2 can be populated with components that make up the machinery for mapping sensory inputs to response actions, as does Russell in [54]. We now discuss some applications of GLAIR that we have been developing.

Some important general features of GLAIR-agents 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 (Sect. 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's physiology, and (c) no reference is made to external objects. The second property of

[20] As we all know, 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. [43]). Elephants don't even stand a chance in this respect.

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[21] 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 touching something hot. 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* [62]. 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 [48, 5]. 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 [62]. Situated action is used in *reactive planning*[2, 16, 55].

Deliberative behavior requires considerable processing of data and reasoning which results in action. In anthropomorphic terms, this is learned behavior that requires reasoning that can be modeled by a Turing Machine (or first order logic), for example explicit planning and action.

[21] E.g., visual reflexes in [52]: Here responses are generated to certain visual stimuli that do not require detailed spatial analysis.

We have developed an implementation mechanism for the Perceptuo-Motor-level which we call Perceptuo-Motor-Automata [28]. 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 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 [28]. A PMA may become active as the result of an intention to execute an action at the Knowledge level [35]. 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 monitored 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 Instance: the Color Labeling Agent

We now present an instance of an agent conforming to our architecture, the Color Labeling Agent (CLA). It has a set of *grounded* or *embodied* concepts represented as terms at its Knowledge Level, a "sub-conscious" color space at its Perceptuo-Motor Level, and a color camera at its Sensory-Actuator Level, as well as a color monitor which it uses as a primitive actuator to point to things in its field of view (Fig. 2).

Using a normalized Gaussian function of perceptual color space coordinates as the basic category model, the CLA is able to

1. *Name* real colors in response to real visual stimuli (camera images), and provide a confidence or "goodness of example" or (fuzzy) membership value.
2. *Point out* examples of named colors in real images and provide a confidence rating, and as a derivative of this capability, pick the *best example* of a named color from a set of color samples, or from an image in general.

The model is at present constrained to the eleven so-called *basic color categories* and their corresponding names in English, as defined first in the work of [8]. The CLA's performance on these tasks has been quantitatively shown to be reasonably consistent with human performance [41]. In particular, this means that

1. The model places the *foci* of the basic color categories in the same regions of the color space as human subjects do.

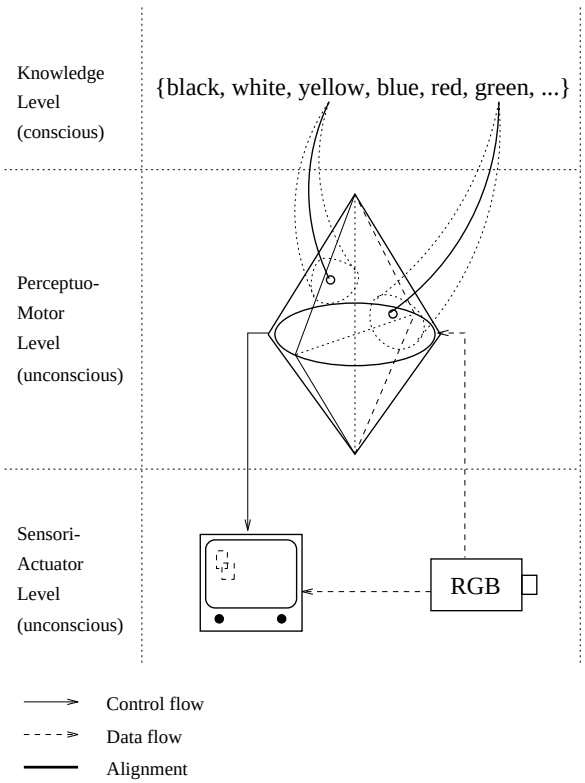


Fig. 2. The various parts of the Color Labeling Agent, relative to the levels of our architecture.

2. The model places the *boundaries* of the basic color categories in the same regions of the color space as human subjects do.

As such, the color categorization application can be seen as a (partly) *embodied* [39, 32, 26] or *grounded* [19] system, or as an instance of *situated cognition* [62].

The current implementation of the CLA consists of two separate parts, one concerned with selection and display of samples from images, and the other concerned with the actual color perception and categorization model (Fig. 3).

The display program runs under X windows, and allows one to display a 24-bit RGB image. It also lets one select samples of a certain pre-defined size (currently 12×16 pixels) from the image using the mouse, which will be passed to the categorization program. It can subsample the entire image using the same blob size, and pass the result to the categorization program. Finally, it can draw boxes around blobs, whose center coordinates it gets from

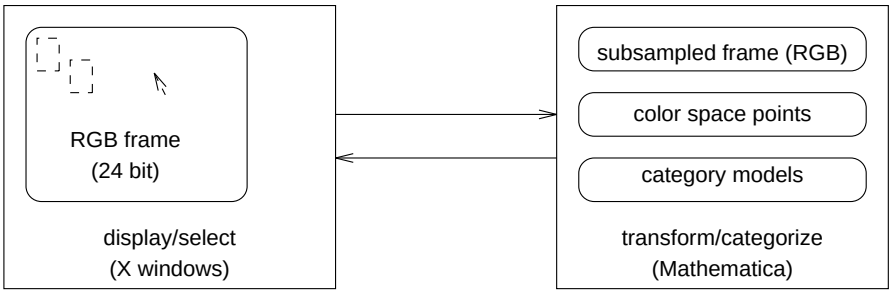


Fig. 3. Outline of the color naming/pointing-out/selecting application, consisting of a display and select part (left), implemented as an X windows client, and a transformation and categorization part (right), implemented in Mathematica code. The two parts communicate asynchronously via a simple file protocol.

the categorization program.

The categorization program is a collection of Mathematica functions that can (1) *Name* the color of a blob pointed to on the image, and provide a membership value, (2) *Point out* examples of a named color category on the image, and provide a membership value, and (3) *Select* one from a number of samples whose color best fits a named category. The names returned can be simple or complex, and the best n candidates can be returned. The category membership threshold θ may be specified, as well as the underlying color space to use. The names specified for the pointing-out function can be simple or complex as well, and the threshold θ can be specified, as well as the color space to use. The function can either return any n examples (the first exceeding the threshold) or the best n examples. The select function always points to the best example of the specified category within the set of samples provided, using a specifiable underlying color space. It will ask the user to provide it with a set of samples from the image first, to get around the absence of any image segmentation or object recognition algorithms.

For a detailed quantitative and qualitative analysis of the performance of the Color Labeling Agent we refer to [41] and [40]. Suffice it here to say that the performance is reasonably consistent with human performance on the same tasks, which means to us that we have succeeded in grounding a small set of elementary symbols in perception and action – although the active side of the agent is relatively primitive.

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 the long term consequences (i.e., beyond the current situation) of its actions.
- The agent is computationally resource bounded. 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 dog-fights. ABS is an interactive video-game where a human player plays against a computer driven agent. The game starts up by displaying a game window and a control panel window (Fig. 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 uses the control panel to choose a move, which is a combination of changing altitude, speed, and direction. When (s)he presses the go button, the computer agent also selects a move. The game simulator then considers both moves to determine the outcome, and updates the screen and the accumulated damage to planes. ABS simulates simultaneous moves this way. If a player's plane is close in altitude and position to the enemy plane, and the enemy is in frontal sight, the latter is fired on automatically (i.e., firing is not a separate action). The levels of damage are recorded in a side panel, and the game ends when one or both of the player's planes are destroyed.

The computer agent has been developed in accordance with the principles of the GLAIR architecture. Figure 5 schematically represents its structure. Initially, the agent does not have any PMAs available, and uses conscious level reasoning to decide what move to make. Once situation transitions are learned and cached in a PMA, the agent uses the PMA for deciding its next move whenever possible. Hence, 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. On trial runs, we started ABS with an empty PMA and as the game was played, transitions of the PMA were learned. Also, when similar situations occurred and there was an appropriate PMA response, the PMA executed the corresponding action. As the game was played, we observed that the

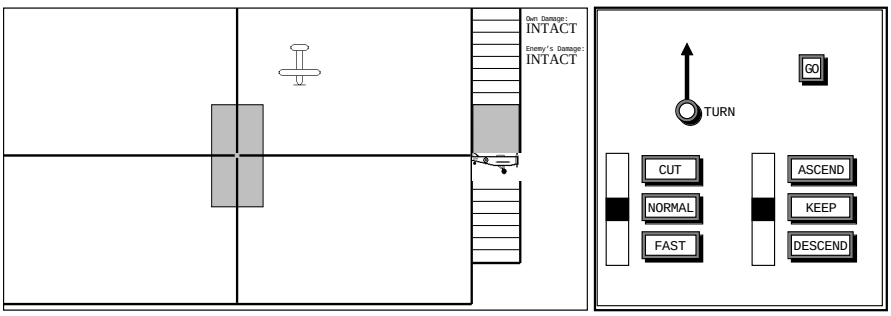


Fig. 4. Air Battle Simulation game window and control panel (see text).

agent became more reactive since the PMA was increasingly used to generate behaviors instead of the Knowledge level.

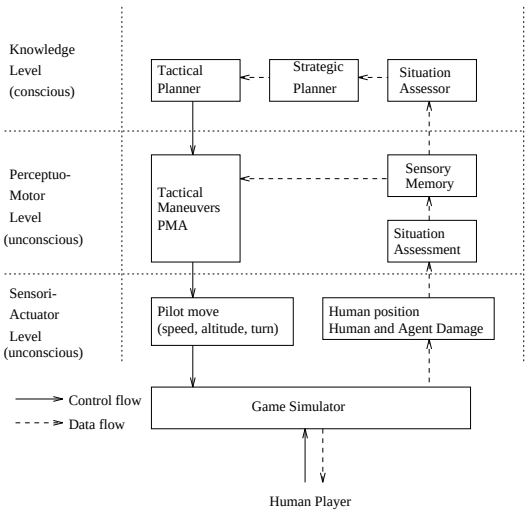


Fig. 5. Schematic representation of the Air Battle Simulation GLAIR-agent.

Improving "Unconscious" Behaviors. The rules of a PMA are situation/action pairs. 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 within firing range. Let's assume that

we can assign a goodness value G(s) to each situation s between -1 and 1. As the pilot makes a move, he finds himself in a new situation. This new situation is not known beforehand 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, his view of the game is not Markovian. Let $\mathbf{Q}$(s,a) be the evaluation of how appropriate action a is in situation s, and $\mathbf{R}$(s,a) be the goodness value of the state that the pilot finds himself in after performing a in situation s. The $\mathbf{R}$(s,a) values are determined as the game is played and cannot be determined beforehand. This is called the immediate reward. We let $\mathbf{Q}(s,a) = \mathbf{R}(s,a) + \gamma \max_k \mathbf{Q}(s',k)$ where situation s' results after the pilot performs a in s. γ is a parameter between 0 and 1 that is known as the discount factor in reinforcement based learning. At the start of game, all $\mathbf{Q}$(s,a) in the PMA are set to 1. As the game is played, $\mathbf{Q}$ is updated. As of this writing we are experimenting with setting appropriate parameters for $\mathbf{Q}$.

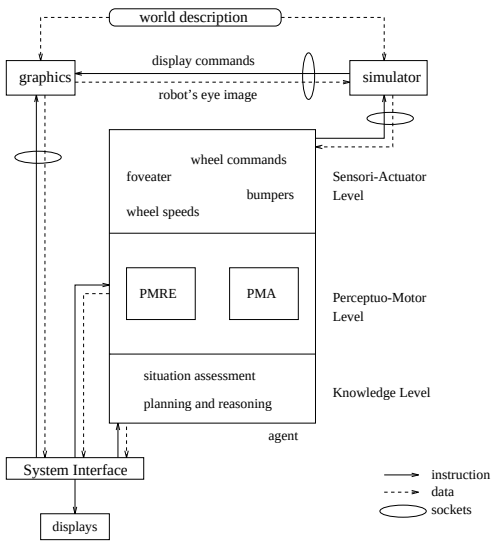
Observing Successful Patterns of Interaction in the World. We assume that the agent does not know about the long term consequences of its actions. Furthermore, the reinforcement based learning we have used assumes a Markovian environment. That is, the agent believes the world changes only due to its 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 the 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 an atomic action at that level [23].

We plan to explore other learning techniques such as experimentation as a form of learning [60]. 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 have developed 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. A complete setup using MRL consists of a GLAIR-agent, a simulator with an incorporated description of a physical environment, and a graphical interface (Fig. 6).

Emergent Behaviors. A major objective for this project is learning emergent behaviors. Like Agre with his improvised actions [2] and Brooks with



MRL system diagram

Fig. 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.

his subsumption architecture [9] 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 minimality and the choice of primitive actions is physiological. In other words, 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.

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 Fig. 7.

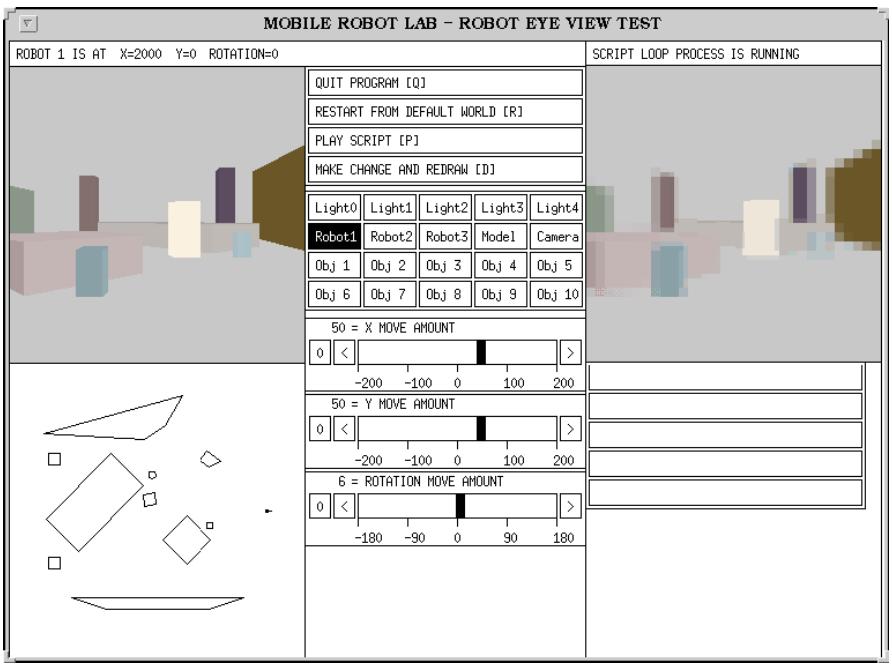


Fig. 7. The graphical interface of the mobile robot simulator. Upper left: color perspective view of the environment, from the robot's point of view. Upper right: same, but in a "rexelated" view from the robot's rectangular foveal camera (see text). Lower left: floor plan of the environment. The robot's current position and orientation is indicated by the small triangle near the right edge, halfway from the top. Middle: interface and simulator control panel, including camera and lighting model, and movement controls. Courtesy of Amherst Systems Inc, Buffalo NY.

The room the robot moves in 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) specifying the surface spectral reflectance (color).

Any number of robots may inhabit the room. They have two independently driven wheels on either side, 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 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 can be simplified in different ways, e.g. using a space-variant downsampling to create a "rexelated" image as used in Hierarchical Foveal Machine Vision [7, 45]. This simplified camera view is computed and passed to the simulator by the graphical interface, on the basis of the 3D perspective views.

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 spectral reflectance function. All objects are assumed to be Lambertian reflectors.

The Mobile Robot Lab has been used to study GLAIR-agent based gaze control in the context of space exploration, in a man-machine cooperative scenario [7], in addition to some other explorative work.

4 Concluding Remarks

We have presented a general architecture for autonomous embodied 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 a few (physical and simulated) agents embedded in their environment, in various stages of implementation. Some additional aspects of our work are discussed in [25], [27], and [29].

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[22] mind makes for an awesome combination.

5 Acknowledgements

This work was supported in part by Equipment Grant No. EDUD-US-932022 from SUN Microsystems Computer Corporation, and in part by NASA under SBIR contract NAS 9-19004. We appreciate comments made on an earlier draft of this paper by Phil Agre, Chris Brown, Stevan Harnad, Donald Nute, John Pollock, Beth Preston, William Rapaport, and Tim Smithers. Goofs

[22] He-man or She-man.

are entirely attributable to the authors, of course. Thanks to Cesar Bandera and Robert J. Makar of Amherst Systems Inc. for kindly providing the Mobile Robot Simulator interface snapshot. Apologies to John Steinbeck, who carries no blame for this paper.

References

1. P. Agre. The dynamic structure of everyday life. Technical Report 1085, MIT Artificial Intelligence Laboratory, MIT, 1988.
2. P. E. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of AAAI-87, Seattle WA*, pages 268–272, July 1987.
3. J. Albus, A. Barbera, and R. Nagel. Theory and practice of hierarchical control. In *23rd International IEEE Computer Society Conference*, pages 18–38, 1981.
4. J. R. Anderson. *The Architecture of Cognition*. Cambridge: Harvard University Press, 1983.
5. S. Anderson, D. Hart, and P. Cohen. Two ways to act. In *ACM SIGART Bulletin*, pages 20–24. ACM publications, 1991.
6. D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs NJ, 1982.
7. C. Bandera, H. Hexmoor, and S. C. Shapiro. Foveal machine vision for robots using agent based gaze control – final report. Technical Report SBIR-NAS 9-19004, Amherst Systems Inc., Buffalo, NY, September 1994.
8. B. Berlin and P. Kay. *Basic Color Terms: Their Universality and Evolution*. University of California Press, Berkeley CA, 1991 edition, 1969.
9. R. Brooks. A robust layered control system for a mobile robot. Technical Report 864, MIT AI Labs, MIT, 1985.
10. R. Brooks. Planning is just a way of avoiding figuring out what to do next. Technical Report 303, MIT AI Labs, 1987.
11. R. A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.
12. S. Card, T. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Erlbaum, Hillsdale, N.J., 1983.
13. D. Chapman. Vision, instruction, and action. Technical Report 1204, MIT Artificial Intelligence Laboratory, MIT, 1990.
14. J. Connell. SSS: A hybrid architecture applied to robot navigation. In *IEEE Conference on Robotics and Automation*, pages 2719–2724, 1992.
15. J. Culbertson. *The Minds of Robots*. U. of Illinois Press, 1963.
16. R. J. Firby. An investigation into reactive planning in complex domains. In *Proceedings of AAAI-87*, pages 202–206, 1987.
17. J. Fodor. *The Modularity of Mind*. MIT Press, 1983.
18. E. Gat. Reliable goal-directed reactive control of autonomous mobile robots. Technical report, Dept. of Computer Science, Virginia Polytechnic Institute and State University, 1991.
19. S. Harnad. The symbol grounding problem. *Physica D*, 42(1-3):335–346, 1990.
20. S. 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.

21. R. Hausser. *Computation of Language: An Essay on Syntax, Semantics and Pragmatics in Natural Man-Machine Communication.* Springer-Verlag, New York, NY, 1989.
22. H. Hexmoor. An architecture for reactive sensor-based robots. In *NASA Goddard conference on AI*, Greenbelt, MD, 1989.
23. H. Hexmoor. Representing and learning successful routine activities. Unpublished PhD Proposal, 1992.
24. H. Hexmoor, G. Caicedo, F. Bidwell, and S. Shapiro. Air battle simulation: An agent with conscious and unconscious layers. In *University at Buffalo Graduate Conference on Computer Science 93 (TR-93-14)*. Dept. of Computer Science, SUNY at Buffalo, New York, 1993.
25. H. Hexmoor, J. Lammens, G. Caicedo, and S. C. Shapiro. Behavior based AI, cognitive processes, and emergent behaviors in autonomous agents. In G. Rzevski, J. Pastor, and R.Adey, editors, *Applications of AI in Engineering VIII, Vol. 2, Applications and Techniques*, pages 447–461. CMI/Elsevier, 1993. Reprint available TR-93-15, CS dept., SUNY/Buffalo.
26. H. Hexmoor, J. Lammens, and S. Shapiro. Embodiment in GLAIR: A grounded layered architecture with integrated reasoning for autonomous agents. In D. D. Dankel, editor, *Proceedings of the 6th Florida AI Research Symposium*, pages 325–329. Florida AI Research Society, 1993.
27. H. Hexmoor, J. Lammens, and S. C. Shapiro. An autonomous agent architecture for integrating "unconscious" and "conscious", reasoned behaviors. In *Proceedings of Computer Architectures for Machine Perception*, New Orleans, LA, 1993. Preprint available as TR-93-36, CS dept., SUNY/Buffalo.
28. H. Hexmoor and D. 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 as TR-92-23, CS dept., SUNY/Buffalo.
29. H. Hexmoor and S. C. Shapiro. Examining the expert reveals expertise. In *Proceedings of the Third International Workshop on Human and Machine Cognition: Expertise in Context (Seaside, FL)*, 1993.
30. L. Kaelbling. Goals as parallel program specifications. In *Proceedings of AAAI-88*. Morgan Kaufman, 1988.
31. L. Kaelbling and S. Rosenschein. Action and planning in embedded agents. In P. Maes, editor, *Designing Autonomous Agents*, pages 35–48. MIT Press, 1990.
32. P. Kay and C. K. McDaniel. The linguistic significance of the meaning of Basic Color Terms. *Language*, 54(3):610–646, 1978.
33. D. Kumar. An integrated model of acting and inference. In D. Kumar, editor, *Current Trends in SNePS–Semantic Network Processing System: Proceedings of the First Annual SNePS Workshop*, pages 55–65, Buffalo, NY, 1990. Springer-Verlag.
34. D. Kumar. *From Beliefs and Goals to Intentions and Actions: An Amalgamated Model of Inference and Acting.* PhD thesis, Technical Report 94-04, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY, 1994.
35. D. Kumar. The SNePS BDI architecture. *Journal of Decision Support Systems—Special Issue on Logic Modeling*, 1994. Forthcoming.
36. D. Kumar and S. C. Shapiro. Acting in service of inference (and *vice versa*). In D. D. Dankel II, editor, *Proceedings of the Seventh Florida Artificial Intel-*

ligence Research Symposium, pages 207–211. the Florida AI Research Society, St. Petersburg, FL, May 1994.

37. 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.
38. J. E. Laird, A. Newell, and P. S. Rosenbloom. SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.
39. G. Lakoff. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind.* University of Chicago Press, Chicago, IL, 1987.
40. J. M. Lammens. A somewhat fuzzy color categorization model. Submitted to ICCV-95.
41. J. M. Lammens. *A Computational Model of Color Perception and Color Naming.* PhD thesis, State University of New York at Buffalo, 1994. Also available as TR-94-26, CS dept., SUNY/Buffalo.
42. P. Langley, K. McKusick, and J. Allen. A design for the ICARUS architecture. In *ACM SIGART Bulletin*, pages 104–109. ACM publications, 1991.
43. H. J. Levesque. Logic and the complexity of reasoning. *Journal of Philosophical Logic*, 17:355–389, 1988.
44. P. Maes. Action selection. In *Proceedings of the Cognitive Science Society Conference*, 1991.
45. R. J. Makar. GLAIR: Mobile robot lab camera simulation. Master's project dept. of Computer Science, SUNY at Buffalo, NY, 1994.
46. J. L. McClelland, D. E. Rumelhart, and G. E. Hinton. The appeal of parallel distributed processing. In D. Rumelhart, J. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing*, chapter 1, pages 3–44. MIT Press, Cambridge MA, 1986.
47. D. McDermott. Robot planning. Technical Report CS-861, Yale University, 1991.
48. D. Payton. An architecture for reflexive autonomous vehicle control. In *Proceedings of Robotics Automation*, pages 1838–1845. IEEE, 1986.
49. J. Pollock. *How to Build a Person.* MIT Press, 1989.
50. J. Pollock. New foundations for practical reasoning. In *Minds and Machines*, 1992.
51. W. J. Rapaport. Syntactic semantics: Foundations of computational natural-language understanding. In J. H. Fetzer, editor, *Aspects of Artificial Intelligence*, pages 81–131. Kluwer Academic, New York, 1988.
52. D. Regan and K. Beverly. Looming detectors in the human visual pathways. In *Vision Research 18*, pages 209–212. 1978.
53. J. K. Rosenblatt and D. Payton. A fine-grained alternative to the subsumption architecture for mobile robot control. In *Proceedings of the International Joint Conference on Neural Networks*, 1989.
54. S. Russell. An architecture for bounded rationality. In *ACM SIGART Bulletin*, pages 146–150. ACM publications, 1991.
55. M. J. Schoppers. Universal plans for unpredictable environments. In *Proceedings 10th IJCAI*, pages 1039–1046, 1987.
56. S. C. Shapiro. Cables, paths, and 'subconscious' reasoning in propositional semantic networks. In *Principles of Semantic Networks.* Morgan Kaufman, 1990.

57. 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.
58. S. C. Shapiro and W. J. Rapaport. The SNePS family. *Computers Math. Applic.*, 23(2–5):243–275, 1992.
59. S. C. Shapiro and the SNePS Implementation Group. *SNePS 2.1 User's Manual*. Department of Computer Science, SUNY at Buffalo, 1994.
60. W.-M. Shen. *Learning from the Environment Based on Actions and Percepts*. PhD thesis, Carnegie Mellon University, 1989.
61. R. Simmons. An architecture for coordinating planning, sensing, and action. In *Proceedings of the DARPA planning workshop*, pages 292–297, 1990.
62. L. A. Suchman. *Plans and Situated Actions: The Problem of Human Machine Communication*. Cambridge University Press, 1988.
63. P. H. Winston. Learning structural descriptions from examples. In R. J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, pages 141–168. Morgan Kaufmann, San Mateo CA, 1985 (orig. 1975).

This article was processed using the LaTeX macro package with LLNCS style