

# An intelligent agent architecture based on the primate brain

Alan H. Bond, Ph.D.,  
California Institute of Technology, Mailstop 256-80,  
1201, East California Boulevard, Pasadena, CA 91125  
email: [bond@cs.caltech.edu](mailto:bond@cs.caltech.edu)  
webpage:<http://www.cs.caltech.edu/~bond>

December 23, 2001

**Abstract:** An information-processing architecture for autonomous intelligent agents is described. Computational principles are derived from the structure and functioning of the primate brain. Based on these principles, an agent architecture is developed. We describe an implementation of an agent and results obtained with multiple interacting agents.

## 1 Introduction

Understanding how to create intelligent autonomous agents that can cooperate is an important challenge. Already, many different agent architectures have been proposed [Bond and Gasser, 1988] [Huhns and Singh, 1998], including SOAR [Laird et al., 1986] and ACT [Anderson and Lebiere, 1998].

In this paper, we report on our research in which we derive a new agent architecture from first principles from the primate brain. We will begin by developing computational principles which are true of the brain and which can be stated quite generally. This is based on an extensive review and study we have made of the the structure and functioning of the primate neocortex [Bond, 2001]. Then, from these principles, we develop a computational approach and architecture. We report on an implementation of intelligent autonomous agents with this architecture. We then describe cooperative and problem solving behaviors obtained with these agents.

## 2 The biological basis of our computational approach

In this section, we examine what we know about the primate cortex, and we develop the basic computational elements from which to design an agent architecture.

**Areas.** The primate cortex is partitioned into distinct areas, each involved in specific kinds of processing and types of data, shown in Figure 1.

**Areas have specific fixed interconnectivity.** The connectivity among areas is the same, or similar, for all primates. Areas are typically connected to a small number of other areas. Connections divide into long range and short range. At short range, an area is often connected to several neighboring areas that are contiguous with it. At long range, an area is usually connected to one, two or three areas that are further away, and not contiguous with it.

**Perception-action hierarchy.** Areas are organized as a *perception-action hierarchy*, shown in Figure 2.

**Processing is distributed.** Areas process data received and/or stored locally by them. There is no central manager or controller.

**There is a uniform process.** The primate cortex has a uniform structure over all of its area [Creutzfeldt, 1978] [Ullman, 1991], having a six layer organization comprising neurons from a small set of anatomical types. The numbers of these cells per unit volume are very uniform over the cortical surface, the main differences being in motor cortex which has more and larger pyramidal cells, and in visual cortex, which has a significantly, three times, greater density of cells. A canonical neocortical circuit can be described. Although long range connectivity, as we have seen, tends to be clustered around cortical regions, short and medium (<3mm) connectivity within the cortex is statistically quite uniform. It therefore appears that information processing within different cortical regions has a common basis or principle.

**Cortical processing proceeds at a uniform rate.** All areas do similar amounts of processing and run at about the same speed. The time to process information in one area and to pass it on to the next area is about 20 milliseconds (Edmund Rolls, personal communication).

**Data parallelism in communication, storage and processing.** We assume that data is coded in parallel codes, such as population codes, so that a large set of parallel fibers carries a code for one message or one meaning. We assume that processing within an area is also highly parallel, operating on a large set of parallel fibers concurrently. Parallel coded data is transmitted, stored, and triggers processing. Processing acts on parallel data to produce parallel data.

**The cortex works in real-time.** The cortex's fastest reaction time to a stimulus is about 100 milliseconds. Further, the path from incoming sensory stimulus to outgoing motor command runs through about five areas. Language is processed in real-time, both generation and recognition, and, as Goodwin has shown [Goodwin, 1981], even the co-construction of a sentence usually occurs in real-time including nonverbal signalling between participants

during the generation of the sentence. Hence the action of an area is, at least some of the time, an immediate reaction to its incoming data. Cognition occurs by areas exchanging data and by repeatedly reacting to incoming data and newly computed data.

**Data is “wide”.** The data items being transmitted, stored and processed can involve a lot of information; they can be complex. Thus, if we have a parallel set of one million neurons, then the code for one choice or component of a data item might involve 10,000 neurons, and then the set of neurons might transmit 100 such components or choices simultaneously as one data item.

**Continuous action.** The cortex is always active, even when calmly scanning under alpha rhythm, or asleep. Further, imaging data show that the change in energy consumption during focussed activity versus general activity is rather small, about 10%.

**Learning occurs in specialized learning areas.** There is some learning in the neocortex, however many types of learning occur elsewhere. Learning of episodes occurs in the hippocampus, learning of routine sequences in the basal ganglia, and affective learning in the amygdala.

### 3 Our agent architecture

**Modules.** We structure an agent as a set of parallel modules with fixed interconnectivity similar to the cortex, and where each module processes only certain kinds of data specific to that module.

**Data items, and their storage and transmission.** We view all data streams and storage as made up of discrete data items which we call *descriptions*. We represent each data item by a logical literal which indicates the meaning of the information contained in the data item. A datatype is taken to be the set of ground literals which unify with a given (nonground) literal. In order to allow for ramping up and attenuation effects, we give every literal an associated weight, or strength, and is a real number. An example data item is `position(adam,300,200,0)` which might mean that the perceived position of a given other

agent, identified by the name “adam”, is given by (x,y,z) coordinates (300,200,0).

**Processing within a module.** We represent the processing within a module by a set of rules. A rule matches to incoming transmitted data items and to locally stored data items, see Figure 3. All the processing by a module is described by a set of left-to-right rules which are executed in parallel.

Rule patterns also have weights, and the strength of a rule instance is the product of the matching data item weights and the rule weights, multiplied by an overall rule weight.

A rule may do some computation which we represent by arithmetic. This should not be more complex than can be expected of a neural net. The results are then selected competitively depending on the data type. Typically, only the one strongest rule instance is allowed to “express itself”, by sending its constructed data items to other modules and/or to be stored locally. In some cases however all the computed data is allowed through.

**Uniform process.** The uniform process is then the mechanism for storage and transmission of data and the mechanism for execution of rules.

**Uniform rate.** We achieve uniformity of rate by describing time by a discrete time scale; the architecture runs in discrete time cycles. In one processing cycle, all the rules in all the modules are executed once, that is, all rule instances, and then all selected data are communicated between modules and/or stored locally.

**Perception-action hierarchy.** Modules are organized as a *perception-action hierarchy*, which is an abstraction hierarchy with a fixed number of levels of abstraction.

The perception hierarchy receives sensory data items at the bottom and derives higher level descriptions to form a percept. The action hierarchy generates more and more detailed descriptions of action, that is, it elaborates the plan to the point where motor actions are generated at the bottom of the action hierarchy.

An example of a perceptual rule is:

```
if position(M,X,Y,Z),orientation(M,A),self_position(X1,Y1,Z1),
```

```
then oriented_towards(M),
provided(angle_towards(X,Y,Z,X1,Y1,Z1,A1),app_equal(A,A1)).
```

i.e., from data giving another agent's position and orientation, and from the subject's own position, calculate the angle from the agent to the subject and test if that angle is the same as the agent's orientation, if so, create a new datum representing the fact that the other agent is oriented towards the subject.

An example of an action elaboration rule is:

```
if plan_self_action(walk_towards(M)),position(M,X,Y,Z)
then plan_self_act(walk_towards(X,Y,Z)).
```

i.e., if the planned action for the self in terms of relations is to walk towards some agent, and if this agent's position is X,Y,Z then generate a new datum, which represents planned action for the self in terms of detailed position, to walk towards the position X,Y,Z.

The goal module has rules causing it to prioritize the set of goals that it has received, and to select the strongest one which is sent to the highest level plan module.

**The external world.** Agents operate in an *external environment*. An agent has *sensors* which interrogate the environment and generate sensed feature descriptions which are represented as literals. These input data items are sent to specified modules each cycle. Some modules act as *effectors* in that they send commands, represented as literals, to the environment. The environment receives commands from all the agents and computes what changes to make. Agents can only communicate with each other via the environment.

**Perception-action dynamics.** A plan is selected and elaborated down the action hierarchy, receiving input from the perception hierarchy to allow it to elaborate appropriately. This differs from related ideas [Neisser, 1976] [Albus, 1981] [Arbib, 1981] [Muller, 1997].

**Conditional elaboration - situation.** Within a given level, the component of the action hierarchy at that level is elaborated down to the next lower level, and evaluations are assessed and transmitted back up to the next higher level. By *elaboration* we mean taking data which

describe action at one level and generating data which describe that action in more detail. More detail includes (1) exactly how to act (which detailed action components), (2) in what order, (3) exactly at what times, (4) exactly where in space, and (5) who will do which actions. By an *evaluation* we mean, for example, a value indicating progress, success or failure; such a value can also be associated with a particular datum, for example, one representing an action or goal.

**Conditional perception - attention.** The perception hierarchy and action hierarchy cooperate closely. The action hierarchy must elaborate the currently selected plan conditionally upon the perceived environment. The modules of the perception hierarchy at a given level derive information required for successful action elaboration at that level. The perception hierarchy receives descriptions representing tuning information and direct requests, attention information, and prediction information, from the action hierarchy. This information provides a context for perception, and enables the optimal use of processing and communication resources by the perception hierarchy in supporting the realtime action. Thus, our perception-action architecture provides a framework for attention mechanisms.

**Continuous action.** Action is continuous with a small time granularity, our implementation runs at about 100 milliseconds on a 300MHz processor. Thus, stored data are updated every cycle, the selection of rule instances is updated every cycle, and updated motor commands are output to the environment every cycle. The process of goal generation, goal selection, plan selection, plan elaboration, action specification and motion specification proceeds continuously, renewing the information every cycle.

**Viable behavioral states.** We developed a notion of *viable behavioral state* which is a distributed set of rule activations in different modules which are dynamically linked together by what we call *confirmation descriptions*. The way confirmation descriptions work is as follows. If a module receives a data item that causes successful activity, it sends a positive confirmation message back to the sender, evaluating that data item and boosting the rule

activation sending that data. This tends to stabilize distributed activity. If, on the other hand, received data does not cause any execution, a negative confirmatory signal is sent back, which tends to attenuate the sending rule activation, and thereby to allow competing choices to be tried.

The basic action of the agent is to try to establish an optimal viable behavioral state, that is, one consistent with the response of the environment and with its own motivations. It does this by trying different alternatives at each level on a competitive basis, and subject to confirmation of successful elaboration.

**Joint action.** We generalized the standard artificial intelligence representation of plan to one suitable for action by more than one collaborating agent. A *joint plan* is a set of joint steps, with temporal and causal ordering constraints, each step specifying an action for every agent collaborating in the joint plan, including the subject agent. The way a plan is executed is to attempt each step in turn, and during a step to verify that every collaborating agent is performing its corresponding action and to attempt to execute the corresponding individual action for the subject agent. We made most of the levels of the planning hierarchy work with joint plans, the lower ones work with a “selfplan” which specifies action only for the subject agent.

**4 Our implemented intelligent agent The initial world.** We developed a multiagent system which was a social group of agents in a 3D spatial world. The social structure was based on affiliative and authority relations. Each agent had a social relations memory module which maintained a record of these relations for the entire social group. This module generated goals to service affiliation and authority relations. These goals were achieved by joint social actions among agents.

**The initial agent.** We developed an initial agent [Bond, 1999b] [Bond, 1999a] consisting of data and process representations, with eight memory modules, shown in Figure 4. An outline of each of these memories, the descriptions they store and the processes they include,

is as follows:

- (i) the *social relations* module contains all affiliative and authority relationship information. It generates affiliation and authority goals and sends them to the *goals* module.
- (ii) the *goals* module contains all goals currently held. It activates the most important goals and sends this information to the *overall plans* module.
- (iii) the *overall plans* module receives goals and instantiates suitable joint-plans, sending them to the *specific joint plans* module.
- (iv) the *specific joint plans* module receives a joint-plan, and generates a detailed action based on descriptions received from the perceptual hierarchy. For the others involved in the joint plan, the detailed action or state is verified, and for the self, its detailed action is sent to the *detailed actions for self* module.
- (v) the *detailed actions for self* module receives the detailed self action from the *specific joint plans* module, receives object and location information from lower levels of the perceptual hierarchy, mainly from the *agent positions and movements* module, and outputs a detailed motor action for this to the *motor* system.
- (vi) the *agent positions and movements* module receives sensory descriptions of the state of the external world and provides information on requested agents to the *agent actions and relations* and *detailed actions for self* modules.
- (vii) the *agent actions and relations* module computes higher-level descriptions of the action of each agent involved in the current joint action. It requests information on particular agents from the *agent positions and movements* module.
- (viii) the *plan agents* module receives information from the *overall plans* module as to which other agents are involved in the joint action, and passes this on to the *agent actions and relations* module.
- (ix) the *motor* system does some processing to generate the external action given the direct action received from the *detailed actions for self* module.

The perceptual hierarchy is simply the *agent positions and movements* and *agent actions and relations* modules, and the action hierarchy is the *overall plans, specific joint plans* and *detailed actions for self* modules.

**Implementation.** We implemented the agents in Sicstus Prolog on a network on Linux boxes, with one agent per machine, and the environment and visualization interface on two other machines. Communication was via TCP/IP and sockets. Results were obtained with affiliation, group dynamics, the social use of space and problem-solving behaviors. These behaviors were obtained using about fifteen rules per module.

## 5 Behaviors investigated, and results

**Joint affiliative action.** Agents were given joint plans to offer and receive affiliative action. Affiliative action was structured into four phases, for the offering initiator - (orientation, approach, prelude, then affiliation), and for the receiver - (waiting, orientation, prelude-response, then affiliation-response), and we developed suitable rules for activity in each module in each phase. We demonstrated a pair of agents establishing and mutually controlling viable joint action leading to affiliation, and to satisfaction of their affiliative goals. Authority can be treated similarly with pairwise trials of ability and strength. In addition, authority is increased by numbers and strengths of affiliative relations, particularly with agents of high authority.

**Group dynamics.** To investigate group behavior and the resolution of conflicts among the goals of several agents, we set up scenarios with four and six agents with social goals which had some conflict. For example, two agents' highest goals were to affiliate with a third agent, which in turn had a goal of affiliating with a fourth agent. We were able to achieve smooth dynamical change of joint action. The initial attempts at joint action would fail by the absence of perception of appropriate action by the intended participating agent. This caused the dismantling of the initial plan and the selection of alternative social goals involving other agents. We also achieved the same scenario with a different agent which

had an additional *social disposition* module for the perception of the dispositions of others. Dispositions were represented as positive or negative evaluations of certain goal types. A disposition represented the subject agent's perception of the attitude of another agent toward a given goal.

**Social use of space.** We have also implemented some social spacing behaviors, so that agents tend to position themselves near others, plan paths that avoid others of higher authority and paths that displace others of lower authority. These behaviors use new modules which include not only a spatial map but also a *social map* which represents space and its social significance. Thus, each agent has a different social map, perceives situations differently, and behaves in space differently.

**Problem solving.** We have implemented three Tower of Hanoi strategies described in Anzai and Simon's work [Anzai and Simon, 1979]. This involved distributing the perceptual tests to the perceptual hierarchy. This also produced perceptual attention control. We are currently investigating the learning of these strategies using an episode learning module.

**Natural language processing.** We are developing modules for natural language generation and recognition, following the lexicalist approach of Kempen [Vosse and Kempen, 2000].

## 6 Discussion

This agent architecture differs from others in fundamental ways:

1. It is distributed, modular and parallel, having no global data or control.
2. Datatypes are important, each module is limited to processing only certain data types.
3. It is based on a uniform process and a uniform rate of processing.
4. Action is continuous, so that input and updating of stores occurs every cycle and all matching rule instances fire every cycle. There are two main types of consequence of this (i) different control structures are possible, for example the satisfaction of integrity constraints on data can be achieved by tests applied during the continuous regeneration of that data, (ii) implementation using RETE methods may not be efficient when all data items are updated

every cycle.

5. An active perception-action hierarchy of the type we describe has many useful properties, including conditional elaboration, attention, situated action and layering of control. Existing search methods do not use perception systematically.
6. Data representations are distributed and multimodal. For example, there can be several different concepts of obstacle - visual, logical and motor - which coactivate.
7. Action proceeds by viable distributed perception-action processes. Multiple agents enter into jointly viable action.
8. It does not conform to Newell's physical symbol system hypothesis [Newell, 1990]. What is generated, communicated and stored are logical descriptions. (i) descriptions can be complex, carrying for example of complete plan or a complete lexical entry. We expect to include sensory data such as 2 1/2 D sketches as (large) descriptions. (ii) there is no notion of symbol as data in the basic architecture. As required, we can define a data type of symbol and define its properties using rules. (iii) the basic operation is the matching (unification) of descriptions.
9. There is a fixed set of modules with fixed connectivity. This means that there is a fixed broad data abstraction hierarchy. Hierarchical perception and planning take place within this arrangement. We can think of this as a "hardware" aspect of the architecture.
10. It is based on the brain.

**Acknowledgements.** This work was supported by the National Science Foundation, Computation and Social Systems Program, Research grant IIS-9812714, and by the Caltech Neuromorphic Engineering Research Center, NSF Research Grant EEC-9730980. The author would like to thank Professors Pietro Perona, Mani Chandy and Steven Mayo for their support and encouragement.

## References

- [Albus, 1981] Albus, J. S. (1981). *Brains, behavior, and robotics*. Byte Books, Peterborough, New Hampshire.
- [Anderson and Lebiere, 1998] Anderson, J. R. and Lebiere, C. (1998). *The atomic components of thought*. Lawrence Erlbaum Associates, Hillsdale, New Jersey.
- [Anzai and Simon, 1979] Anzai, Y. and Simon, H. A. (1979). The Theory of Learning by Doing. *Psychological Review*, 86:124–140.
- [Arbib, 1981] Arbib, M. (1981). Perceptual Structures and Distributed Motor Control. pages 1449–1480. in [Brooks, 1981].
- [Bond, 1999a] Bond, A. H. (1999a). A System Model of the Primate Neocortex. *Neurocomputing*, 26-27:617–623.
- [Bond, 1999b] Bond, A. H. (1999b). Describing Behavioral States using a System Model of the Primate Brain. *American Journal of Primatology*, 49:315–388.
- [Bond, 2001] Bond, A. H. (2001). An Information-processing Analysis of the Functional Architecture of the Primate Neocortex. *Journal of Theoretical Biology*, submitted.
- [Bond and Gasser, 1988] Bond, A. H. and Gasser, L. (1988). *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers, San Mateo, CA.
- [Brazier and Petsche, 1978] Brazier, M. A. B. and Petsche, H. (1978). *Architectonics of the Cerebral Cortex*. Raven Press, New York.
- [Brooks, 1981] Brooks, V. B. (1981). *Handbook of Physiology, Section 2: The Nervous System, Vol. II, Motor Control, Part 1*. American Physiological Society.
- [Creutzfeldt, 1978] Creutzfeldt, O. D. (1978). The neocortical link: Thoughts in the generality of structure and function of the neocortex. pages 357–383. in [Brazier and Petsche, 1978].

- [Goodwin, 1981] Goodwin, C. (1981). *Conversational organization : interaction between speakers and hearers*. Academic Press, New York and London.
- [Huhns and Singh, 1998] Huhns, M. N. and Singh, M. P. (1998). *Readings in agents*. Morgan Kaufman, San Mateo, California.
- [Laird et al., 1986] Laird, J., Rosenbloom, P., and Newell, A. (1986). *Universal subgoaling and chunking : the automatic generation and learning of goal hierarchies*. Kluwer, Dordrecht.
- [Muller, 1997] Muller, J. P. (1997). *The design of intelligent agents: a layered approach*. Springer-Verlag, Berlin. Lectures Notes in Computer Science 1177.
- [Neisser, 1976] Neisser, U. (1976). *Cognition and reality : principles and implications of cognitive psychology*. W. H. Freeman, San Francisco.
- [Newell, 1990] Newell, A. (1990). *Unified theories of cognition*. Harvard University Press, Cambridge.
- [Ullman, 1991] Ullman, S. (1991). Sequence-seeking and Counter Streams: A Model for Information Processing in the Cortex. AI Memo 1311, Massachusetts Institute of Technology, Artificial Intelligence Laboratory.
- [Vosse and Kempen, 2000] Vosse, T. and Kempen, G. (2000). Syntactic structure assembly in human parsing: a computational model based on competitive inhibition and a lexicalist grammar. *Cognition*, 75:105–143.

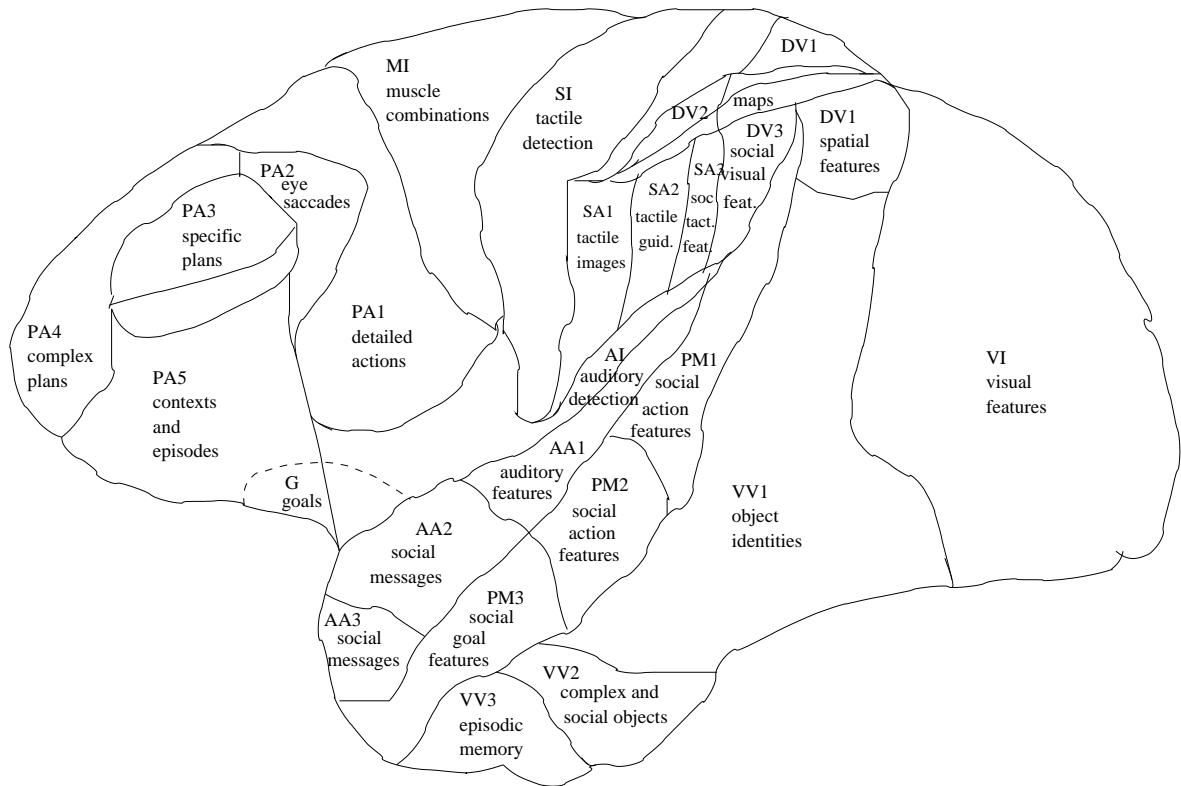


Figure 1: Lateral view of the cortex showing regions and functional involvements

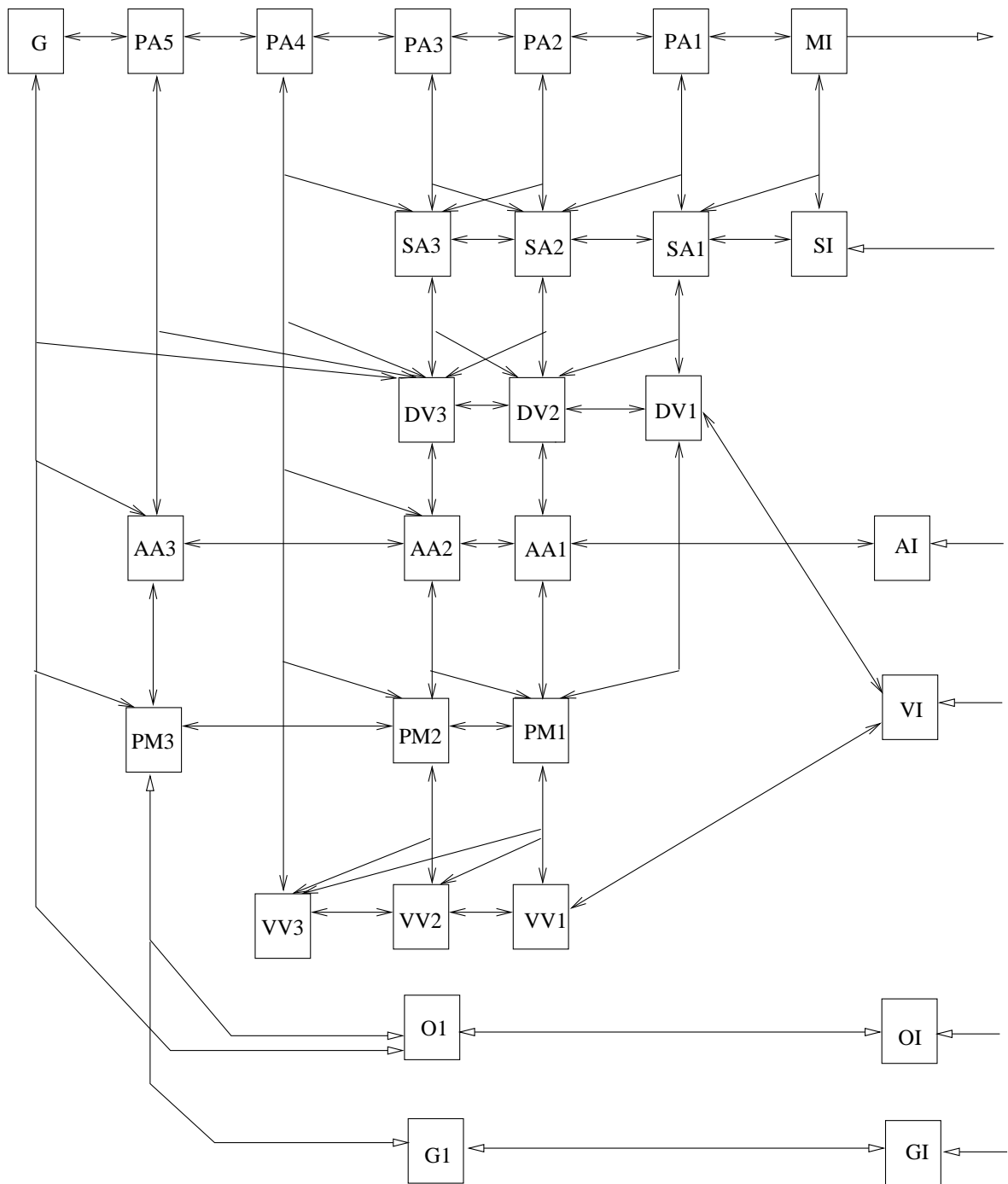


Figure 2: Perception-action connectivity of areas, top of hierarchy to the left

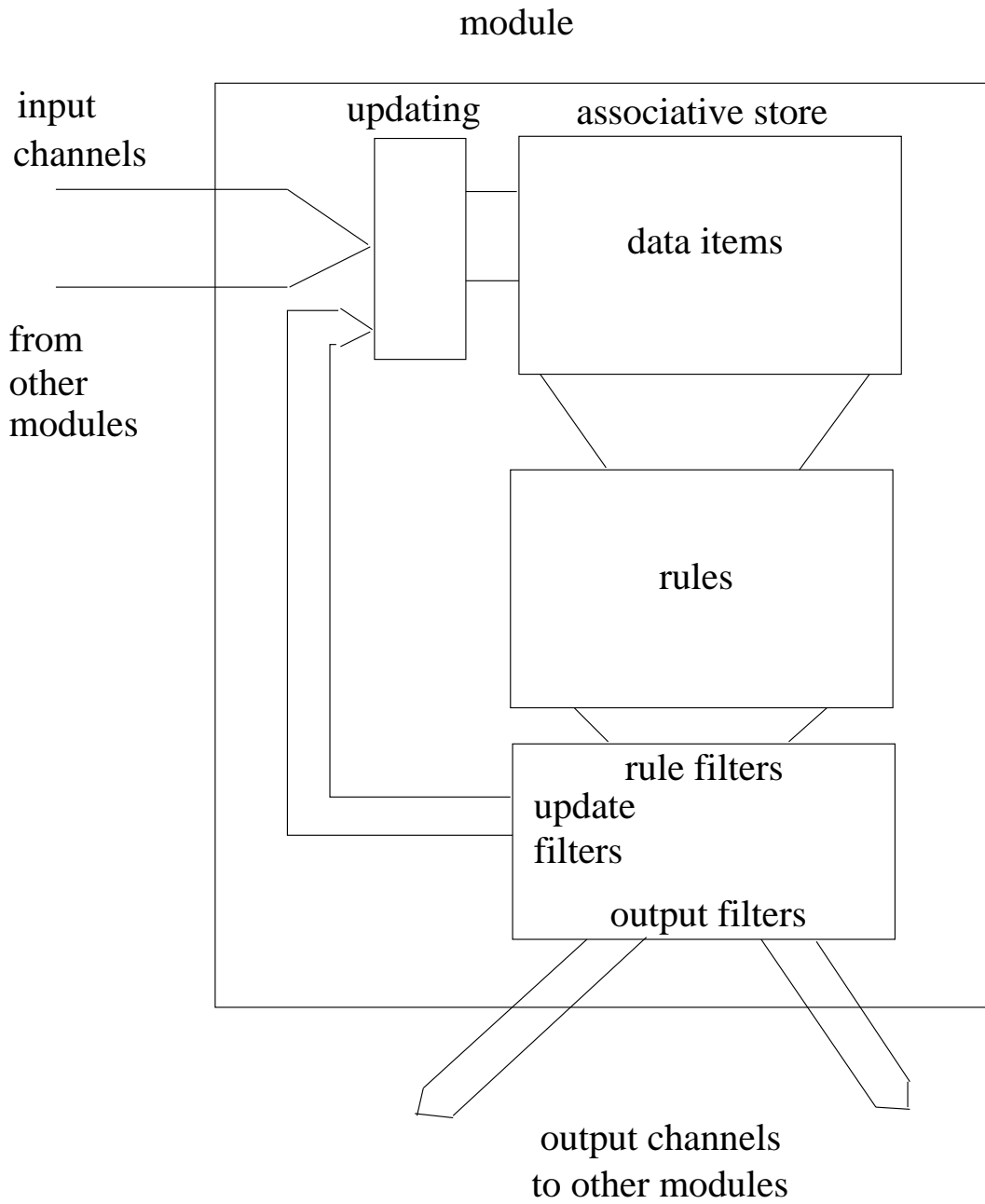


Figure 3: Structure of a module

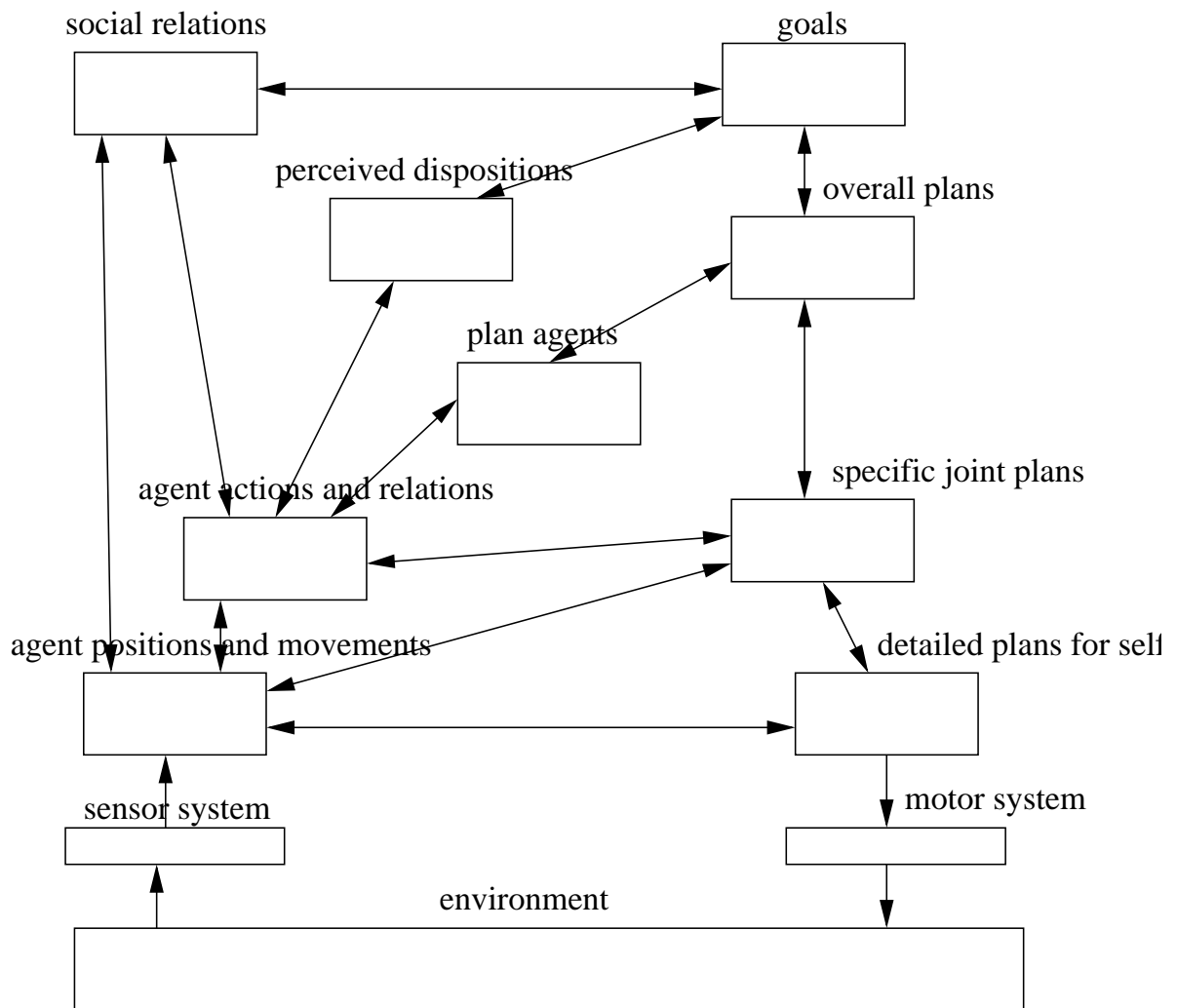


Figure 4: Our initial system model