

Behavior of an adaptive ANN-AI system working with cues and competing concepts

Csaba Szepesvári¹

András Lőrincz

Department of Photophysics, Institute of Isotopes of the
Hungarian Academy of Sciences

Budapest, P.O. Box 77, Hungary H-1525

E-mail: lorincz@obelix.iki.kfki.hu

Fax: (36)-1-156-5045

¹János Bolyai Institute of Mathematics

Attila József University of Szeged, Szeged, Hungary, H-6720

E-mail: szepes@obelix.iki.kfki.hu

April 28, 1995

Abstract

The concepts are presented of a neural model based shell that integrates artificial neural networks (ANN) and artificial intelligence (AI) for problem solving. The shell may possess inherited and learnt ANN and AI subsystems. The shell has and develops (i) cues to the environment for dimensionality reduction, (ii) rules between elements of the reduced dimensional internal representation, (iii) 'concepts' for achieving goals, i.e. for solving existing problems, (iv) the shell then causes the concepts to compete in order to come to a decision. The shell is designed for control problems, e.g. robotic tasks, control of plants, investment advisory systems, and may have very different ANN and AI parts. Here, we consider a simple robotic-like object in two dimensional space.

Keywords: adaptivity, artificial neural network, artificial intelligence, self-organization

Running title: ANN-AI integration

1 Introduction

The history of developments in artificial intelligence (AI) systems goes back a long way (Bundy, 1990). This is also true of research on artificial neural network (ANN) models; such research goes back at least to the original work of Hebb (1949). Ever since their universal approximator nature was proven (Hornik, Stinchcombe and White, 1989), there has been a growing interest in ANN systems. Just a few years ago AI and ANN concepts were considered to be different. Recent papers, however, build ANN models that solve AI problems (Thagard, 1989, Peng and Reggia, 1989). It is, in fact, hard to make a clear distinction between AI and ANN systems. One might try to separate them by saying that AI systems are rule based systems, a collection of if-then rules. However, the simplest neural system is a receptor neuron connected to a motoric neuron and that similarly works as an if-then rule: if we have the stimulus from the environment then the receptor neuron activates the motoric one and we have the reaction to the stimulus.

One might insist and try to consider every artificial neural network as an AI system. From this point of view one might consider the perceptron as an if-then rule with a decision surface: if the perceptron's n dimensional input is above a decision hypersurface then the output is 'high', otherwise the output is 'low'. Although the decision surface concept may be formulated with AI methods it does not fit AI procedures. The underlying reason is that AI systems are not designed for continuous and noisy input problems, they are designed to and can handle the sequential rules of a simplified world. Even then sequentiality has its own traps. On sizing up the problems one meets combinatorial explosion in computation time or in memory requirements. ANN systems, on the other hand, have no combinatorial explosion but can barely handle sequentiality. Let

us consider the game of chess as an example. There are 32 figures in chess with around 100 possible moves in one iteration. This is to be compared with the image of chess on the retina that has more than 1,000,000 "pixels" (rods and cones), color and intensity variations, therefore a much larger degree of freedom. If we take a numerical example considering six steps ahead the number of variations is 100^6 , and this is to be compared with the degree of freedom of the recognition problem, which is in the order of $10^{1,000,000}$ assuming only a ten grade gray scale. The processing time in chess, however, is not limited by the recognition of the table but by the recognition of the situation, the possible outcomes, in other words the sequentiality. Sequentiality thus seems to have a different working mechanism in the brain, and we shall design it with the help of AI.

This is our starting point, or recipe, in the design of AI and ANN integration: take self-organizing ANN (i) for categorization, (ii) for learning the topology of the external world, i.e. for sizing down the problem to a set of 'high' or 'low' bits. It is in this reduced 'internal representation' that AI could try to work by searching for rules and for sets of rules.

In the work we present here the AI generates concepts that fit previous experience. AI, however, is not capable of choosing from the possible list of concepts to generate a response. It is not capable since AI systems were designed to list possible solutions and are not capable of making decisions between solutions of the list or, even worse, if the problem has no satisfactory solution. In other words the shell needs a decision stage at the very end. The stage we have developed for making decisions is closely linked with Thagard's ECHO system (Thagard, 1989). The decision then influences both the environment and our object. The changes in the external world and in the internal parameters are

then detected with the self-organizing ANN's. This last step leads to new rules, to the overwriting of old rules, or it leaves the AI system unchanged.

2 Self-organizing cues to the environment: Dimensionality reduction

If one says that 'a complex environmental stimulus has triggered a response' it means that (i) there is a complex environment that can and did stimulate the system, (ii) there was a simple or complex response to the stimulus and (iii) the response was triggered by the stimulus. The word triggering means that a switch inside the system was set to 'high' and it launched the response. It means that the switch has an input system, a cue, that is stimulated if it fits the external world and the switch has an output, an internal representation of reduced dimensionality of the external world that is set 'high' if it matches the stimulus or is 'low' if it does not. This is a reflex type of response, and the switch is the simplest representation of the external world. If a switch is set high then we say that the said elementary feature is present. If this elementary feature triggers an immediate response then we call it a reflex and it could be an inherited property of the system. If it does not require an immediate response then it may be used for further processing for later responses. This cue concept to the external world 'if it fits then a switch is set high' (Cloak, 1975) is the place where an AI system can operate because the dimension is already reduced and there is time for processing.

The first building block of our shell is, then, an input system that provides outputs for further processing and reduces dimensionality. This dimension reducing system depends on the task one tries to solve.

If there are stable features in the external world then, for example, a cat-

egorizing system may serve well. The categories could be inherited or learnt. If inherited, then adaptivity is severely decreased. For category learning only self-organizing artificial neural networks are suitable since there is no supervisor who could pair inputs to outputs. We would like to mention two self-organizing networks, the two layer ART network of Carpenter and Grossberg (1975) that sets up categories with the help of behavioral success and the Hebb-anti-Hebb network of Földiák (1991) that has a constant drive for distinguishing inputs and forms memory saving sparse representations from them.

If there is a topology in the external world, then the topology might be wired into the ANN as in Kohonen type networks, the network otherwise being self-organizing. The topology itself may not be known a priori. The model of Szepesvári, Balázs and Lőrincz (1992) provides spatial filters for position estimation of extended objects in a self-organizing fashion. The object of the external world excites the receptors of a high resolution ‘eye’ and the neurons develop spatially localized filters and relations between these filters. The self-organized filters and the relations between them discover the topology of the external world, and allow applications like learning collision-free robot arm movements. Spatial filters are useful as a means of reducing the high resolution image to a few positions.

As shown by these examples, self-organizing ANN’s have inherited properties that help the learning process if the external world fits the information built into the network, but otherwise limit performance. The more information the self-organizing ANN is built into, the less adaptivity it has; on the other hand the less the inherited information the longer the learning procedure. We consider self-organizing learning as the slow learning process of an infant (Hebb, 1949). The building up of the AI system that works on the outputs of the dimension

reducing self-organizing ANN's may be taken as maturity learning. As stated by Hebb (1949) "The prompt learning of maturity is not an establishing of new connections but a selective reinforcement of connections already capable of functioning."

3 Building concepts of the AI system

From the system's point of view, the world is made of two parts: the internal world, i.e. the system itself, and the rest of the world – the environment. This distinction is a very delicate one and we shall not try to define it. We would mention, however, that when writing the software it is clear that the food is part of the environment but the mouth is part of the system. However, from the point of view of operation there is no need to make this distinction, we need a different categorization. From the operational point of view we need three categories for the internal representation of the world:

- Internal representation of objects, whether they are internal or external. The internal representation of external objects has been considered in the previous section. Internal representation does not only consist of the representation of external objects, but we include the representation of internal objects giving such information on hand, foot, mouth, etc. Bits represents objects or features of objects. We shall call the set of bits of the internal representation at a given instant an *I-state*.
- Operators of the system. We assume that the system can influence the environment, can move, may eat, grasp, etc. This assumption means that the system has operators associated with the interaction; thus it may have an operator for 'moving forward' or for 'lifting its foot', or 'opening its mouth' or just for 'contracting a muscle'. Every operator is associated

with a state: if the state is set ‘high’ then the operator is activated, if it is set ‘low’ then the operator is inactive. We shall call the set of operators at a given instant an *O-state* .

- Goals of the system: these are special states that can be either ‘high’ or ‘low’ that initiate search procedures in the I-state and the O-state to find routes to fulfil them. We might say that: the system is in a ‘neutral state’, ‘it is hungry’, ‘it is interested’, ‘it is suffering’, etc. All of these may be formulated with the help of hunger: it is hungry for food, it is hungry for information or it is hungry to escape from the present situation. This hunger type of formulation is general enough for it to be used in very different fields, such as robotic motion (hunger for quick and collision-free access), economy task solving (hunger for net income), optimization problems (hunger for making compromises), etc.

The AI system is built under the following principles: The I-states are the ones the AI should work on. The task is then to experience *I-O-I* time sequences. The *I-O-I* triplets that were encountered contain all the information the system can collect, and the task is to process this information in order to fulfil the goals. The place where systems might differ is their strategy for ‘subtracting rules’ to simplify or speed up search procedures, to generalize from examples, etc. One might try to use a truth maintenance system (Doyle, 1983), or something else, but it should be non-monotonic logic. It is the environmental feedback that validates the set of collected ‘rules’. The rules may be used to plan future actions. This is the point however, where the limitations of the AI systems on feature extraction from ‘noise’ restrict generalization. It means that planning could be made better by making a distinction between important and accidental features. Here we deal with the *I-O-I* triplets only, and do not investigate the

problem of generalized ‘rules’.

It is worth mentioning that if one is collecting the *I-O-I* triplets for a given operator *O*, one cannot expect the influence of operator *O* to be predicted with certainty. The reason for this is that the internal representation has restricted information about the whole world. In mathematical form: operators are not mappings on the *I*-state space, but relations. Traditional AI systems are based on the assumption that *I-O-I* triplets are given, and it is these that determine the operators. Here, the restricted information on the operators means that the full effects of the operators are not known and may never be known. As a result we have not used TMS systems, since TMS stores the last consistent set of rules only. If one saves only consistent sets in the erroneous learning stage, important information may be lost.

4 Concepts, and the competition of concepts

This system was designed for problem solving. We formulated problems as goals. Goals may be achieved by *I-O-I* series. If an *I-O-I* series is found that promises the achievement of a goal, then we say we have a first concept to match the goal. In practice the actual concept is the organizing of *I-O-I* triplets. If we have a single concept then we proceed accordingly. However, the system might develop a set of concepts. In this case one has to decide what concept ought to be followed. An attractive way of solving this problem is the suggestion of Csányi and Kampis (1985) on the competition of concepts. The ECHO system of Thagard (1989), although it comes with a very different motivation, may be considered as an implementation of the competing concepts. Here we refer to the literature and will not describe ECHO in detail, especially since the competition in our system is somewhat different. Details of the algorithm are given in the

next section.

5 Implementation of the model

In this section we describe a possible implementation of the shell. It has three major parts: the interface system, the cognitive system, and the decision system. The implementation is quite general: it is the interface system and the set of inherited rules of the cognitive system that are the only problem dependent parts of the shell. The example we present here corresponds to a simplified robotic like object. The adaptive part of the cognitive system, and the decision system are independent of this special example. For this robotic like object (for expedience we shall call it *Robjct*) we simulate the environment, consisting of objects causing pain and objects allowing energy refilling. It is easy to define hunger in this case: *Robjct* is hungry if its energy is low. *Robjct* will be programmed to avoid painful situations.

In the first subsection we describe in detail the environment, the interface system and the rules of the interaction between the environment and *Robjct*. The second subsection provides details about the cognitive and decision systems.

5.1 *Robjct*'s world

We assume that *Robjct* has well developed self-organizing eyes. The role of these eyes is to provide position and category information about the objects in the neighborhood of *Robjct*. Several attempts have been made in the field of ANN to solve this problem. For self-organizing category learning see, for example, the ART model of Carpenter and Grossberg (1987), and the sparse representation of Földiák (1991). For position estimation we have made an attempt that makes use of extended objects (Szepesvári et al., 1992). Networks that are capable of solving categorization of not well positioned objects have

been presented by Olshausen, Anderson and Van Essen (1992) and Fukushima (1992).

The eyes serve as a means of reducing the high dimensional image of the external world to a few high or low bits. We assume that Robject is living on a grid, and any object of the external world – just as is Robject – is placed on a certain grid point. The world is considered slow compared to Robject’s motion, in other words the objects are static. At any grid point only a single object or Robject may be present simultaneously. Robject is watching the world through self-organized spatial filters (Szepesvári et al., 1992). These spatial filters match the grid size of the external world and Robject can see the neighboring grid points. In the example we are treating, Robject could see 5×5 points or positions. To every position there belongs one of three categories: the empty position, food at that position, or a certain blockage at that position.

Robject’s world is shown in Figure 9. Robject is represented by the sign ‘>’. It means, that Robject is at grid point $X = 2, Y = 2$ and is facing right (Orient 1). Robject’s mouth is at the leading edge of the face (not shown). The ‘Suck’ operator should be ‘high’ to acquire food. Another condition for eating is that the food should be in front of Robject. Food is denoted by ‘@’. If eating is successful then energy is refilled to 1.00. In other instances energy is constantly decreasing. Robject can move forwards or backwards on the grid, and can turn left or right. Robject cannot enter a grid point if it is occupied by food or by blockage (‘#’). Any trial results in pain. Robject’s eye is a ‘chameleon’ eye, Robject can see its 5×5 grid environment. Robject is in the center of the environment. A vector of 24 components represents Robject’s environment. Empty positions are shown as ‘-’. Robject always has one goal as goals have priorities. The list of goals on order of priority is: remove pain,

fill energy, explore. Robject explores the world by activating an operator and collecting *I-O-I* triplets. The operator Robject used in its seventh moment of evolution was the ‘Fore’ operator to move forward.

5.2 Working mechanisms of Robject’s mental architecture

Figure 2 shows the architecture of Robject. The eye (a), the dimension reducing system (b), the internal representation (c), the collection of stored *I-O-I* triplets (d), the goal system (e), and the set of operators (f) have been described. The arrows in Fig. 2 show the flow of information between subsystems. Light and other effects excite receptors. The high dimensional receptor representation of the inputs is connected to the self-organizing dimension reducing ANN’s, and is transformed to a low dimensional internal bit representation. The rest of the system is supposed to work parallel.

The parallel part has three main components, shown in Fig. 3: the cognitive space (CS) that contains the short term memory (STM), the decision system (DS) and the goal system (GS).

The GS is the very core of the whole system. It is preprogrammed not adaptive and determines the behavior of Robject . The working mechanism of Robject may be expressed as learning from successful and unsuccessful shorter or longer time sequences. The inherited properties of GS are: (i) it is able to designate goals, (ii) it is able to determine if the planned goal is achieved, or (iii) it is able to detect pain.

The basis of CS is a dynamic set of directed I-O-I triplet chains. The set is built and erased by the STM of CS. The STM chains the time sequence of consecutive *I-O-I* triplets in a FIFO (first in first out) finite length buffer. The STM is controlled by GS: (i) if the actual goal is fulfilled, then STM tries to lengthen the successful *I-O-I* triplet chain of CS, strengthens the connections of

the chain and associates the actual goal to the triplets, (ii) if the goal was not achieved after the planned chain of actions then STM weakens the connections of the chain, (iii) if the chain of actions leads to pain at any step, then the last *I-O-I* triplet is communicated to CS and STM is erased. Strengthening and weakening mean adding or subtracting 1 from the actual connection strength. Connection strengths are limited to positive values. On reaching zero, the connection disappears.

Another rule was implemented for evaluating *I-O-I* triplets. Every *I-O-I* triplet has a *significance value* (*SV*). At the first appearance of a triplet the *SV* is set to 0.5. Having detected an $I_1 O I_2$ triplet, triplets of initial state I_1 and operator O are selected and their *SV*'s are changed in accordance with the update rule: If the final state of a selected triplet was I_2 then its $SV = SV(I_1 O I_2)$ was increased to \sqrt{SV} otherwise it was decreased to SV^2 . The *SV* set were kept above a certain positive threshold by dropping triplets having *SV*'s smaller than the threshold. This procedure with thresholding could be considered as competition between triplets. This procedure tends to select the two most important triplets for a given *IO* pair. For this small problem and for our sparsely-packed space this is an appropriate choice. More demanding environments and larger memory allocation possibilities allow more general choices.

It is an AI system that works on the dynamic set of CS. In order to describe this AI processor, let us suppose that the dynamic set of CS is not empty. For the sake of quick orientation we briefly sketch a simplified version of our CS's working mechanism:

The directed *I-O-I* statements of CS, and a starting statement and a goal statement are suitable for AI search procedures. AI could search for *I-O-I* lists that reach the given goal. Then the set of *I-O-I*

lists could be passed to DS.

The need for memory and time saving led us to a slight modification of this working mechanism. Figure 4 illustrates the working mechanism. First the set of initial *I-O-I* triplets that may correspond to the actual situation should be chosen. The reason for starting from more than one initial *I-O-I* triplet that may correspond to the actual situation originates from the following:

- different operators may apply to solve a given situation,
- the effect of an operator is ambiguous in the internal representation,
- the *generalizing ability* of Robject may be increased by noting that very similar situations may be equivalent from the point of view of a given goal.

The zeroth step of our procedure was to select a set of initial *I-O-I* triplets called a *kernel*. This step may occur in two distinct cases:

1. a previous goal was reached and a new goal was just chosen, or the DS – to be described below – chose a new operator in a random fashion and tried it if there were unsatisfactory possible outcomes.
2. Robject presumes that it is on the route to reach the goal and the DS has just chosen an *I-O-I* triplet

In the first case a set of initial *I-O-I* triplets is selected whose first I-state is closer to the actual I-state than a heuristically chosen Hamming distance. The same method is followed in the second case if there are no outgoing connections from the chosen *I-O-I* triplet. Otherwise the kernel is shifted to the *I-O-I* triplets at the end of the outgoing connections.

The next step is to find possible outcomes starting from this kernel. Two distinct ways seem to be appropriate: (i) a search for memorized, connected,

successful *I-O-I* triplets (see Fig.4), (ii) a search for unconnected *I-O-I* triplets whose initial state matches one of the final states of the *I-O-I* triplets in the kernel (not implemented here). This procedure could be continued up to timeout. The selected part of CS will be known as *extended kernel*. DS keeps working on the growing extended kernel. We simulate that DS and CS work parallelly.

The decision system (DS) is based on the idea of Csányi and Kampis (1985) on the competition of concepts generated in cognitive space. We have constructed the competition in a way similar to Thagard's ECHO system. The ECHO system has two parts: (i) builds up coherence relations between hypotheses, here these are I-O-I triplets; (ii) utilizes a Grossberg type competitive algorithm (1968) for decision making. In our case coherence relations correspond but are not proportional to connection strengths. Connection strengths are inversely proportional to the degree of diversity, i.e. to the number of possible choices towards the goal. This plays a normalizing role. The principle of "explanatory coherence" of Thagard is not applicable in our case because at the time of decision the goal is not necessarily part of the extended kernel. The algorithm is described in detail below.

We number the *I-O-I* triplets by i . The i^{th} triplet is $T_i = I_1^{(i)} O^{(i)} I_2^{(i)}$. When creating the kernel or at every step of extending the kernel the *external inputs* of the new *I-O-I* triplets are considered as proportional to the closeness of the goal, the significance value of the *I-O-I* triplet, and a balance function A_i , that has positive and negative expectations:

$$E_i = G(T_i) SV(T_i) A_i. \quad (1)$$

Here G is a triple valued goal function: it is 1 if its argument reached the actual goal at least once, it is 0.5 if its argument was an intermediate step to reach

the actual goal once, and 0 otherwise, $SV(T_i)$ is the significance value of the i^{th} triplet, and A_i is the balance function. The balance function differs (i) if the triplet T_i is in the kernel or (ii) if it is not. If triplet T_i is in the kernel then A_i weighs (i) how close the initial state of an $I-O-I$ triplet is to the actual state, (ii) if the final state of the $I-O-I$ triplet has pain, (iii) if the $O^{(i)}$ operator acting on the actual state is dangerous. If triplet T_i is not in the kernel, but predicts the goal, then E_i and thus A_i should be positive unless triplet T_i leads to pain:

$$A_i = \begin{cases} \epsilon_1 d(I_{act}, I_1^{(i)}) - \epsilon_2 P(I_2^{(i)}), -\epsilon_3 P_{prob}(I_{act}, O^{(i)}) & \text{if } T_i \in \text{kernel}; \\ -\epsilon_2 P(I_2^{(i)}) + \epsilon_4, & \text{otherwise} \end{cases} \quad (2)$$

where $\epsilon_i, i = 1, \dots, 4$ are positive constants, $D_{ip}(I_a, I_b) = 1 - d(I_a, I_b)$ is the normalized inner product of its arguments and thus d is a distance like quantity, I_{act} is the actual state, P is equal to 1 if its argument has pain, otherwise it is 0. P_{prob} is the expected pain probability of the operator $O^{(i)}$ when acting on I_{act} . We estimate it in the following fashion: We make a list (\mathcal{L}) of $I-O-I$ triplets where the initial I-state is closer than θ to the actual I_{act} state measured with the help of d , the final state has pain with no regard to the operator. Then we give P_{prob} as the quotient:

$$P_{prob}(I_{act}, O^{(i)}) = \frac{1}{|\mathcal{L}|} \sum_{I_1 O I_2 \in \mathcal{L}, O=O^{(i)}} d(I_1, I_{act}) \quad (3)$$

Decision is made with the help of competition. Competition happens in iterative steps. At the zeroth step the activations (a_i) of the nodes ($I-O-I$ triplets) are set to the E_i external inputs. In one iteration the update rule of the activations is as follows:

$$\Delta a_i = \begin{cases} -\alpha a_i + (1 - a_i) J_i i, & \text{if } J_i > 0 ; \\ -\alpha a_i + a_i J_i, & \text{otherwise} \end{cases} \quad (4)$$

where J_i , the net input to node i is:

$$J_i = \beta \sum_{j \neq i} w_{ij} a_j \quad (5)$$

and w_{ij} is set to $1/k$ if the number of connections from node i is equal to k and nodes i and j are connected. We call this the diversity principle. This iteration could run parallel with the rest of the system. Updating happens at every kernel update.

Decision is made at time-out, when the highest activity node of the kernel is chosen. If the activity of that node is higher than a certain threshold, then the triplet is accepted, otherwise a random choice is made of a situation-allowed new operator. In the special case of Robject the operator ‘move forward’ has a higher probability than the others. This ‘inherited’ property speeds up Robject’s learning as it turns less frequently and explores larger spaces. The selected operator is then tried and the *I-O-I* list is updated.

6 Experimental results

We ran two sets of experiments. In the first one Robject’s world was a 5×5 grid with ‘periodic boundary conditions’; this means that if one leaves the world downwards one comes back from the top and similarly on the two sides. Robject could see the whole world with itself in the center. Having eaten new food was generated at random positions. Otherwise the space was empty. Robject’s energy was decreased by 0.1 at every step. In case of eating it was set to 1.0, the maximum value. If it decreased to 0.3, Robject became hungry and started to search for operations to cease its hunger. Robject felt pain if it hit the food. To consume the food Robject had to face it, stand in front of it, and had to suck.

Figure 5 shows the ratio h/t , where h denotes the number of cases when Robject was hungry, and t denotes the total number of cases. The theoretical minimum value of h/t is equal to 0.34, if the food is generated at random

positions. To reach the theoretical minimum a set of 23 concepts should be collected. The curve shows that Robject's performance improves as time goes on. It almost reaches the theoretical minimum.

Figure 6 shows the number of *I-O-I* triplets collected during learning as a function of time. In these runs strong competition of concepts was applied. In every 500 steps and for every *I-O-I* triplet an outgoing connection was randomly selected and discarded. Then concepts having *SV*'s smaller than 0.1 or having no connections – but not having pain in either of its states – were also eliminated. For 21 carefully chosen *I-O-I* triplets Robject could theoretically reach the 0.34 value. For strong competition it collected approximately 27 rules and performed well, it reached a 0.41 value. In the case of 'weak' competition, when the low connection number *I-O-I* triplets are not discarded the number of rules still grows steadily after 50000 time steps.

In another set of experiments Robject was placed in a 'maze' (see Fig.9). Figure 7 shows Robject's results in the 'maze'. Food was always generated at a given position. The theoretical minimal value for performance was 0.37 with the number of rules being 144. Robject collected 157 rules and reached a 0.51 value.

7 Conclusions

The combination of self-organizing cues to the environment, rule searching non-monotonic AI, and competition of 'concepts' may be considered as an useful adaptive scheme to integrate AI and ANN techniques for problem solving.

8 Acknowledgements

We are grateful to Drs. Klára Konrád and Iván Futó for helpful discussions.

9 Figure captions

- Figure 9. *Robobject's world*

The 5×5 grid shows Robobject (>), food (@), and blockages (#). Information under the grid describes Robobject's full situation.

- Figure 2. *Scheme of Robobject's architecture*

Eye (a), dimension reducing system (b), internal representation (c), collection of stored *I-O-I* triplets (d), goal system (e), set of operators (f). Arrows show the flow of information between subsystems.

- Figure 3 *Working schematics of Robobject*

The 'parallel' mechanism has three main components, the cognitive space (CS) that contains the short term memory (STM), the decision system (DS) and the goal system (GS). The internal representation (IR) and the set of operators (O) are also shown.

- Figure 4 *Extending the kernel*

Black dots denote the *I-O-I* triplets Robobject may choose from. This set is called the kernel. The kernel is extended by considering the corollary *I-O-I* triplets shown by empty circles. Crossed circles denote *I-O-I* triplets reaching the goal.

- Figure 5. *Performance vs. time for an empty space*

The ratio h/t is shown vs. time. h denotes the number of cases when Robobject was hungry, t the total number of cases. The horizontal line shows the theoretical limit of performance.

- Figure 6. *Number of collected rules vs. time*

- Figure 7. *Performance vs. time for a non-empty space*
- Figure 8. *Number of collected rules vs. time for a non-empty space*

References

- [1] A. Bundy, editor. *Catalogue of artificial intelligence techniques*. Springer-Verlag, Heidelberg, 3rd edition, 1990.
- [2] G.A. Carpenter and S.A. Grossberg. Massively parallel architecture for self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 37:54–115, 1987.
- [3] F. T. Jr. Cloak. Is cultural ethology possible? *Human Ecology*, 3:161–182, 1975.
- [4] V. Csányi and Gy. Kampis. Autogenesis: the evolution of replicative systems. *Journal of Theoretical Biology*, 114:303–321, 1985.
- [5] J. Doyle. *Some theories of reasoned assumption: an essay in rational psychology*. Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1983.
- [6] P. Földiák. Learning invariance from transformation sequences. *Neural Computation*, 3(2):194–200, 1991.
- [7] K. Fukushima. Character recognition with neural networks. *Neural Computing*, 4:221–233, 1992.
- [8] S.A. Grossberg. Some nonlinear networks capable of learning a spatial pattern of arbitrary complexity. *Proceedings of the National Academy of Sciences*, 59:368–372, 1968.
- [9] D.O. Hebb. *The organization of behavior*. John Wiley and Sons, New York, 1949.

- [10] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.
- [11] B. Olshausen, C. Anderson, and D. Van Essen. A neural model of visual attention and invariant pattern recognition. *to be published*, 1992.
- [12] Y. Peng and J.A. Reggia. A connectionist model for diagnostic problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(2):285–298, 1989.
- [13] Cs. Szepesvári, L. Balázs, and A. Lőrincz. Topology learning solved by extended objects: a neural network model. *submitted to Neural Computation*, 1992.
- [14] P. Thagard. Explanatory coherence. *Behavioral and Brain Sciences*, 12:435–467, 1989.

@		#		
	#			
	#	>		
	#			
	#			

Figure 1: Robject's world

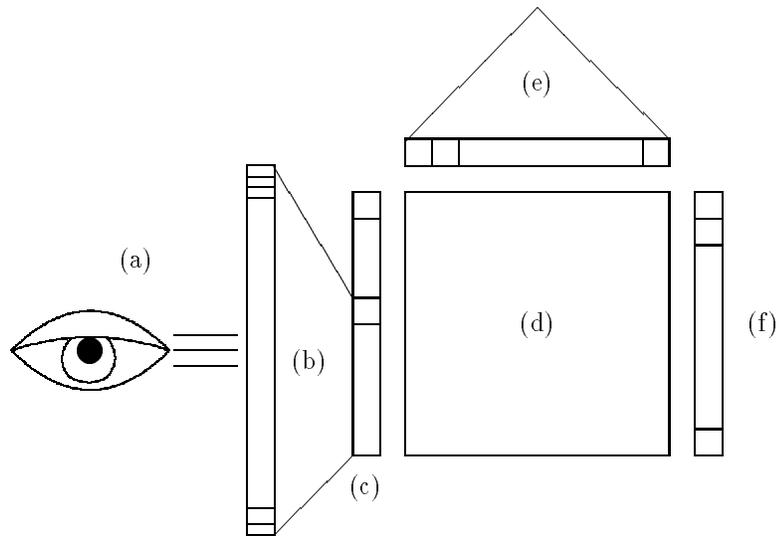


Figure 2: Scheme of Robjet's architecture

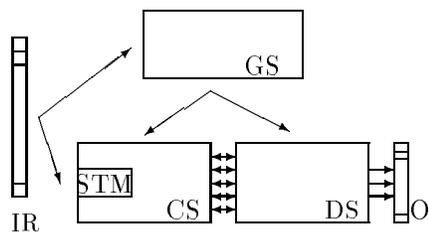


Figure 3: Working schematics of Robject

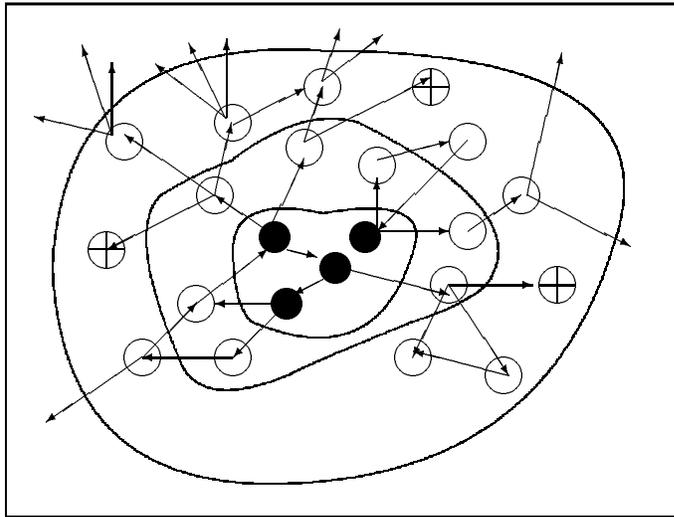


Figure 4: Extending the kernel

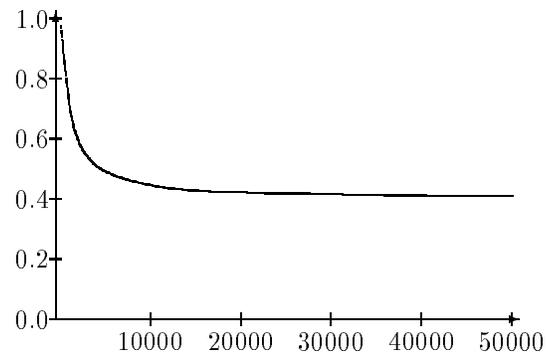


Figure 5: Performance vs. time for an empty space

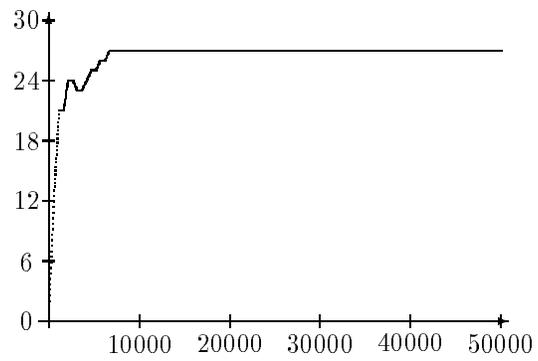


Figure 6: Number of collected rules vs. time for an empty space

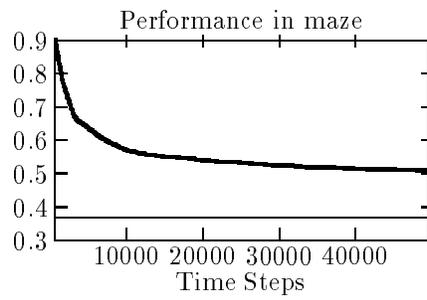


Figure 7: Performance vs. time for a non-empty space

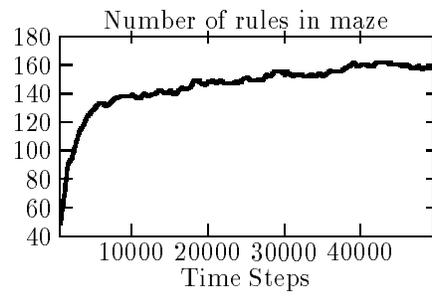


Figure 8: Number of collected rules vs. time for a non-empty space