NEURAL MECHANISMS FOR SELF-ORGANIZATION OF EMERGENT SCHEMATA, DYNAMICAL SCHEMA PROCESSING, AND SEMANTIC CONSTRAINT SATISFACTION

Revised Version

Christian Balkenius

Lund University Cognitive Science
Kungshuset, Lundagård
S–223 50 Lund
Sweden
E-mail: Christian.Balkenius@fil.lu.se

Abstract. The concept of schema and some general characteristics of models using schemata are discussed. It is shown by computer simulations how a combination of a number of simple neural circuits are capable of performing actions similar to those commonly attributed to schemata, especially self-organization of a representational code, recognition of spatial and temporal structure, adaptive performance and semantic constraint satisfaction.

1. INTRODUCTION

What are the basic units of cognitive processing? The so called Classical Theories (Fodor & Pylyshyn, 1988) argue that these units are symbols together with symbolic processes. On the other hand, the Connectionist School argues that we should approach cognition at another level and study how neuronlike elements interact to produce collective effects (Rumelhart et. al. 1986). Both camps seem to assume that the other is totally wrong. Yet, there is no doubt that human behaviour exhibits examples of both the symbol processing capabilities of the Classical School and the interactive activation type of processing in Connectionist models. Nor is it controversial that, on the lowest level, cognitive abilities are implemented as a neural system. A marriage of the symbol processing capabilities of the Classical Theories with the constraint satisfying capabilities of the Connectionist Theories is something that is highly desired. The simplest way to do this is to assume that the cognitive system is supplied with both a classical module and a connectionist module. This approach, that has been adopted by Smolensky (1987), does not answer any questions, however. The issue is not to decide whether the human cognitive system is a symbol processing automaton or a connectionist network (or both), but to establish what level is appropriate for the description of a cognitive system.

1.1 WHAT IS THE CORRECT LEVEL OF DESCRIPTIONS?

Ideally one should start at the neural level and gradually increase the level of abstraction to dispatch from the complexity of the lower levels. Eventually, we would end up with a whole set of descriptive models, each of which could be reduced to the one below. Which model we would use should then depend on the phenomena we are trying to explain. In most cases, our model should be located somewhere on the path from the low level neural model to the higher level cognitive model. This is a very important issue as we try to describe highly complex phenomena.

The different systems that are capable of explaining these phenomena are usually complex enough to be reduced to each other. For instance, it is easy to show how to implement a classical architecture in a neural net, and even easier to go the other way. In fact, every computer simulation of a connectionist model is an example of this. Our task would be easy if we had a complete description of the nervous system, and the processes in it. Unfortunately, what we have is almost nothing more than design principles and some general ideas about structure. Nor do we

Lund University Cognitive Studies – LUCS 23 1993. ISSN 1101–8453.

know what the higher level systems should be. The assumption of the Classical School that the higher level system is a symbol processing automaton can be as wrong as the idea that interactive activation is the basic process in the high level system as well as in the low level system.

1.2 SCHEMATA AS A VEHICLE FOR HIGH-LEVEL PROCESSES

To bridge the gap between the low-level neural models and the high-level symbolic models I will concentrate on the concept of a schema. Schemata are neutral in respect to the different views of cognition and have been used in models from both camps. I want to show how schemata with highly complex structure can self-organize in a neural circuit. With schemata as an intermediate level, it is possible to describe symbolic processes as an interaction among symbol schemata. Note that I am not trying to implement a classical architecture. The schema level is not a symbol manipulating automaton, but have the ability to self-organize in a way that allows behaviour that is usually associated with symbol manipulation as well as the behaviour typically attributed to connectionist models. As the schema level is nothing more than a possible interpretation of a set of neural models, I believe that it is a good candidate for an intermediate descriptive level. In this paper, I will show some possible neural instantiations of the schema system. However, a large set of neural mechanisms could be responsible for the behaviour described here. Many details of the neural model could be changed without losing the emergent properties of the schema level.

1.3 OVERVIEW

The rest of this paper is divided into three sections. The first one discusses some of the properties and problems of various views on schemata. Especially, self-organization, indexing, multiple instantiation and schemata representing temporal structures such as actions, plans and linguistic representations are considered. The second section develops a neurally based model of schema processing that suggests an unified basis for all schema mechanisms. Finally, I present a few preliminary computer simulations that show some of the properties discussed in sections 2 and 3.

2. SCHEMATA

The basic building unit in many theories of cognition is the schema. Even though the concept seems to have as many definitions as authors, some common core exists in all of them. I will use the term schema as a collective name for the structures as used by Piaget (1952, 1973), Arbib and Hanson (1987) and Rumelhart et. al. (1986c). I would also like to include concepts usually denoted by other names such as Frames (Minsky, 1987), Scripts (Schank & Abelsson, 1977) etc. Among the different authors we find some common characteristics of schemata. The following list, adapted from Rumelhart et. al. (1986c) summarizes some of the main features.

1. Schemata are used for representing objects, situations, and actions. This implies that schemata must have dynamic properties.

2. Schemata have variables. This indicates that there are ways to change a schema to adapt it to different situations.

3. Schemata can be embedded. One schema can have another schema as a part or as an instantiation of a variable.

4. Schemata support default assumptions about the environment. They are capable of filling in missing information. Schemata are not passive data structures. They are active representations.

5. Schemata are closely connected to memory.

2.1 SCHEMATA AND CONNECTIONISM

Rumelhart et. al. (1986c) argue that schemata are sets of microfeatures and show how they can emerge from the collective behaviour of small neuronlike elements. Each microfeature is represented by a neuronlike unit that interacts with the others by activating or deactivating them. This interpretation of schema lends itself to implementation in constraint satisfying networks (Ackley et. al., 1985, Hinton, and Sejnowski, 1988, Smolensky, 1988, Hopfield 1982, 1984), and can also account for the self-organization of schemata as the network interacts with its environment. This is a highly desired property of any model of cognition. For Piaget, this was one of the main issues. His view of schemata is more complex and I will only consider a few general aspects of it. Most

important to notice is that schemata, in Piaget's view, are related not only to static representations of concepts, but to actions and temporally organized events as well. This is something that has been missing in most connectionist modelling. Some efforts have been made to handle temporal structure (e.g. Elman, 1989, Grossberg 1986), but, as far as I know, no clear connections to schema theory has yet been pointed out. Neither are connectionist models good at representing multiple instantiations of a schema. For example, Rumelhart and Kawamoto (1988) make the very strong assumption that a schema for a sentence contains exactly four slots. One for each of the possible roles in the sentence. As a result, it is possible to have exactly four instantiations of different schemata at a single time. Of course, we can postulate any number of slots for the number of roles we need in our sentence comprehension model. However, such a solution will always be *ad hoc* and if we want to model discourse comprehension, this is not an accessible path at all, unless we are willing to accept slots for instances like 'the-actor-of-the-second-relative-clause-of-the-third-sentence'.
Such an approach is clearly absurd.

Another problem with slots is how generalisation from one slot to another should be made. Since the corresponding nodes in all slots are not generally active at the same time, the usual learning mechanisms that affect nodes that are currently active, such as Hebb's rule (Hebb, 1949), cannot be used. If slots are used, it is hard to retain systematicity. The main problems of the connectionist account of schemata involves the representation of multiple instantiations and temporal sequences in perception and action. Below, possible mechanisms for the solutions to these problems are discussed.

2.2 COMPUTATIONS IN THE STYLE OF THE BRAIN

Arbib, Conklin and Hill (1987) take a view at a higher level by stating a few added assumptions about schemata. The framework is described as being "in the style of the brain" and their idea of schemata is similar to that of Rumelhart's in some respects but ideas from traditional semantic nets are also used (Woods, 1975). The environment is represented by an assemblage of schemata, each of which corresponds to an object in the environment. Schemata have two sources of activation. A schema receives an input from the environment that corresponds to how well the schema fits perceived objects or events. It also receives activation or deactivation from other active schemata.

One assumption of Arbib's is that schemata can be understood as "master copies" of certain programs. They can be copied and tagged to form multiple instantiations of a schemata. Schemata interact in the same way as the nodes in connectionist networks. This assumption sharply contrasts with the view that a neural assembly is responsible for the representation of a certain schema. Although many parallels to neural mechanisms are used to support this schema theory, no account is given for how schemata can be copied in a neural system. In fact, it is inconsistent with a straightforward neural interpretation of the theory. The copy of a schema has to be associated with a corresponding copy of interaction links to other schemata. In the neural interpretation, this would require the network to copy the actual physical synapses between neural cells. This is a biologically very implausible mechanism. Such a view of schemata cannot be considered to be a neural theory. However, it does raise one important question: how do we represent multiple instantiation of the same schema at a neural level?

2.3 FRAMES AND SCRIPTS

Minsky's notion of *frames* is yet another instance of a schema theory at a higher level (Minsky, 1987), with inspiration from both neural theory and semantic nets. If we look at other higher level accounts for schemata, such as Schank's *scripts*, we see more rigid structures. Schemata are typically considered to be a set of variables and procedures. When we give the variables values, we get an instantiation of a schema. This is also the definition of schemata usually used in the field of Artificial Intelligence (cf. Charniak & McDermott, 1985).

3. A NEURAL NETWORK BASED MODEL OF SCHEMA PROCESSING

The following sections develop a neural network based model of schema processing capable of self-organization of schemata with the properties discussed above. Schemata can be of two types. I will call the first one *static* schemata. They are used to represent static entities such as objects. The other kind of schemata will be called *dynamic* schemata. They are responsible for the representation of actions, plans and processes. A schema theory must account for both these kinds of schemata.

The model incorporates a few basic neural modules, each of which can be implemented in a number of ways. First I develop a mechanism for representing static schemata. This architecture is based on an idealized model of cortical organization. Temporally organized schemata are handled as sequences of static schemata. A special type of chunking module is developed for recognition, prediction, and production of schema sequences.

3.1 AN IDEALIZED MODEL OF CORTICAL ORGANIZATION

The model is based on a few general observations of cortical organization and 'design principles' of the nervous system that together can account for the processing of static schemata. First, the lateral interaction between cortical neurons is usually described as having the form of a Mexican hat (Kohonen, 1988). Neurons close to each other send excitatory signals to each other and neurons further away exchange inhibitory signals. The strength of the signals are functions of the distance between cells and decreases with distance. Second, over larger distances, the cell populations communicate with each other through intracortical association fibres and with other parts of the brain through specific afferent and efferent projection fibres. (cf. Carpenter, 1984). Third, there exists some evidence for division of cell populations into antagonistic pairs (Grossberg, 1987).

If we combine these principles and add some further assumptions, we are able to construct a model of cortical processing. Consider a set, $F^{(a)}$, of neural populations,

$$F^{(a)} = \{v_0, v_1, ..., v_n\}, \quad (1)$$

with the following characteristic (figure 1):

A. Antagonistic clusters

Each population is divided into two antagonistic clusters v_i^+ and v_i^- that have exclusively inhibitory connections to each other. This is a result of short range inhibitory connections. Denote the activity, or short time memory trace (STM), of each cluster by x_i^+ and x_i^-. Because of the inhibitory connections, the activity of the population as a whole stays approximately constant. Thus, we have,

$$x_i^+ + x_i^- \approx C. \quad (2)$$

The result of this, in the case of binary activation values, is that either x_i^+ or x_i^- is on. Networks with this property are often called *dipoles* and have been thoroughly discussed by Grossberg (1978).

B. Excitatory groups

Each population, v_i, is further divided into a set of smaller neural groups,

$$v_i = \{v_{i0}, v_{i1}, ..., v_{in}\}. \quad (3)$$

Each group v_{ij} receives excitatory connections from three sources.

C. Excitatory lateral interaction

The first source is the other neuron groups in the same population. This excitation serves to make the activation of all nodes in the population approximately the same. If we denote the activation of each neuron group in v_i by $x_{i0}, x_{i1},..., x_{in}$ then,

$$x_{ij} = x_{ik}, \text{ for } i,j = 0, 1,..., n. \quad (4)$$

This type of lateral excitation has been used by Kohonen (1988) to acount for the formation of a cluster of active neurons and by Edelman (1989) to describe what he calls neuronal groups.

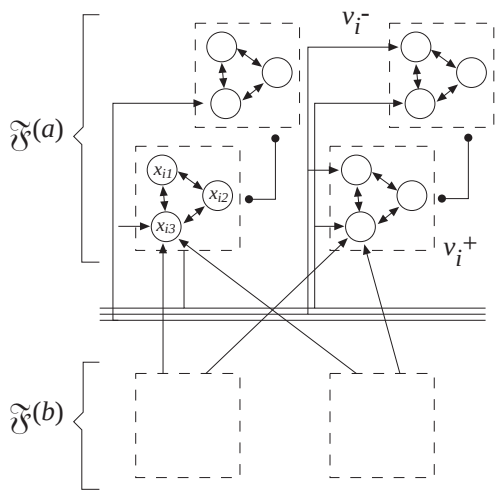

Figure 1. An idealized model of cortical organization. The text describes the architecture in detail.

D. Intracortical Associations

The second source of activation is different for each neural group in v_i. Let group v_{ij}^+ and v_{ij}^- receive input from population v_j^+. Note that only positive populations are allowed to send intracortical associations to other populations, but both positive and negative populations receive excitatory signals from other populations.

E. Projections

The last kind of excitatory connections project from another set of neural populations $F^{(b)}$. All neurons in a population are most responsive to approximately the same activity pattern in $F^{(a)}$. Thus, we can consider each population in $F^{(a)}$ to be a feature detector for activity patterns in $F^{(b)}$. The projection from $F^{(b)}$ to $F^{(a)}$ corresponds to the specific afferent projection fibres to cortex from lower structures or to longer intracortical association fibres.

F. Activity equations

Each population v_i^+ and v_i^- obey the following non-linear equations,

$$\mathbf{F}(dx_i;dt) = -Ax_i+(B-x_i)I\backslash S(_+;i)-\gamma x_i I\backslash S(_-;i), \quad (5)$$

where A describes the passive decay of the activity, B is the upper bound for the activity, I^+ describes the excitatory and I^- the inhibitory input.

$$I\backslash S(_+;i) = \backslash \mathbf{I}\backslash su(_{j=o};^n; f(x_j)w^{+;ji})+I(b)^{;i}, \quad (6)$$

and,

$$I\backslash S(_-;i) = \backslash \mathbf{I}\backslash su(_{j=o};^n; f(x_j)w\backslash S(_-;ji)) \quad (7)$$

where w_{ji}^+ is the strength of the connection from population v_j^+ to v_i^+ and w_{ji}^- is the strength of the connection from population v_j^+ to v_j^-. $I_i^{(b)}$ describes the excitatory signals from $F^{(b)}$. (5) - (7) are an approximation of the collective behaviour of all cells in the population. It is also possible to use a simpler linear approximation of the activity equation (5).

$$\mathbf{F}(dx_i;dt) = -Axi + I\backslash S(_+;i) - I\backslash S(_-;i), \text{ if } x_i \in]0, B[, \quad (5')$$

$$\mathbf{F}(dx_i;dt) = min\{0, -Ax_i + I\backslash S(_+;i) - I\backslash S(_-;i)\}, \text{ if } x_i = B,$$

$$\mathbf{F}(dx_i;dt) = max\{0, -Ax_i + I\backslash S(_+;i) - I\backslash S(_-;i)\}, \text{ if } x_i = 0.$$

For these equations to be valid we have to adjust the w_{ij}:s to compensate for processes in the individual cells in the population. How this is done will be described in the next section.

The output function is chosen to be sigmoid or S-shaped. A typical choice is,

$$f_{ij}(x) = \backslash F(max(0, x-\phi_{ij})^{\lambda};max(0, x-\phi_{ij})^{\lambda}+d), \quad (8)$$

where ϕ is the threshold for the population; λ and δ are positive constants. Output functions similar to (8) have been studied in different contexts in many neural models (e.g. Grossberg 1973, Hopfield, 1984). In many studies the following functions have been used instead,

$$f_{ij}(x) = 1/(1+e^{-x}), \text{ or} \quad (8')$$

$$f_{ij}(x) = tanh(x). \quad (8'')$$

Given the weights w_{ij}, the equations (5)-(8) define a constraint satisfying network. The properties of these equations are more thoroughly discussed by Grossberg (1973).

There are two different ways to understand what happens in each population v_i. The first approach is to look at all the signals that are received by the population and consider what pattern of activity that makes the group of neurons respond. We may consider the neuron group a recognizer or categorizer for certain patterns in the rest of the network. This is what Grossberg (1978) calls an in-star (figure 2). The second approach is to investigate what other populations are activated by v_i. In this case, we say that v_i reads out a pattern in the other populations. The population v_i is called an out-star.

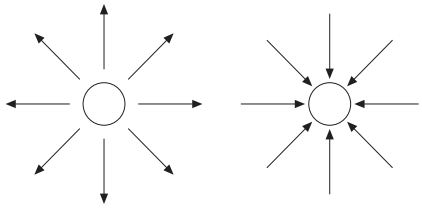

Figure 2. (LEFT) A neuronal out-star; (RIGHT) A neuronal in-star. The two figures illustrate two interpretations of a neuronal operation. In the in-star the neuron reacts on an input pattern while the out-star produces a pattern of activity.

What makes this model presented in this paper different from most connectionist models is not the equations that govern the transient behaviour of the system, but the way in which the synaptic weights that define the processing are defined and changed. This will be presented in the next section.

3.2 NEURAL PLASTICITY

As a result of different activity patterns presented in $F^{(b)}$, the synaptic weights within $F^{(a)}$ are changed to let the system build a model of its environment. The learning process is based on a few simple principles: change of threshold., activity strengthening and environment sampling.

A. Change of threshold

The threshold of each neuron group v_{ij} changes over time according to the following equation:

$$\mathbf{F}(d;dt)\phi_{ij} = -C\phi_{ij} + Dx_{ij} \quad (9)$$

This makes ϕ_{ij} approach the exponentially moving average of x_{ij} (Kohonen, 1988). If the two positive constants C and D are chosen sufficiently small and x_{ij} is an observation from a set of patterns with static distribution, ϕ_{ij} is close to the mean activation of v_{ij}. This can be interpreted as growth of the neurons in v_{ij} if they receive many signals. The larger size of the neuronal soma reqiures more external activation if the activity is going to reach the point where it starts to fire. If the neurons receive few signals, the process is the opposite. It decreases its size and will be easier to activate.

B. Activity strengthening

As the somata of the neurons in v_{ij} grows larger or smaller according to (9), they are more easily activated. The output level of the population, tends to change accordingly. This can be described by,

$$out(x_{ij}) = (\mathbf{F}(1;1-\phi_{ij}))(f(x_i)z_{ij}-\phi_{ij}) \quad (10)$$

if we assume that $\phi_{ij}<1$. The choice of this particular output function makes the range of the output independent of the probabilities of each activity patern in $\mathsf{F}^{(a)}$. The normalization factor $(1-\phi_{ij})^{-1}$ is not absolutely necessary for the system to work but improves the performance considerably.

C. Environment sampling

When the group v_{ij} is active, it samples its environment in order to reproduce it the next time it is activated. This is governed as,

$$\mathbf{F}(d;dt)z_{ij} = x_{ij}[-ez_{ij}+hx_i], \quad (11)$$

where z_{ij} is the strength of the excitatory synapse from x_i to x_{ij}. in analogy with (9), z_{ij} approaches the mean activation of x_i at periods when x_{ij} is active (Kohonen, 1988).

D. A probability approximation of the network equations

If we assume the populations to take on binary values, equations (8)-(11) suggest an approximation for the connections, w_{ij}, described above. This approximation rests on the probability of coactivation of different neuron groups:

$$w_{ij} \approx max(0,\mathbf{F}(p(j|i)-p(j);1-p(j))) \quad (12)$$

$$p(j|i)=p(f(x_j) > \varphi \mid f(x_i) > \varphi), \quad (13)$$

$$p(j)=p(f(x_j) > \varphi). \quad (14)$$

The parameter φ is used to decide when the activity of a neuron is sufficiently close to 1 to be considered active. This rests on the assumption that the network receives binary activation patterns. However, since the neurons normally do not reach their maximal activation, we can not replace the condition $f(x)>\varphi$ with the more obvious $f(x)=1$.

Equations (5)-(14) characterize a self-organizing constraint satisfying network. This can be easily understood if we consider equation (12), which lets us interpret the weights in terms of the involved probabilities. This has four important consequences.

First, the weight of the connection from population v_i to v_j is 0 if the activities of the two populations are uncorrelated. This captures the idea that the activity of one of two uncorrelated populations does not contain any information about the state of the other population. As a result, they should not be able to influence each other's state.

Second, the connections have the same sign as the correlation. This implies that the signs of the two connections between two populations have the same sign. For example, if the activity of population v_i is positively correlated with another population v_j, this population must be positively correlated with v_i. However, the strength of the correlation may be entirely different.

Third, if v_j is always active when v_i is, (i.e. if $p(j|i)=1$) the connection takes on its maximal value. Similarly, if v_j is never active when v_i is, (i.e. if $p(j|i)=0$) then the connection takes on its minimal value. Consequently, there will be strong inhibitory clusters among populations that are never active at the same time and strong excitatory clusters among populations that are always active together. It seems that this property can be used to automatically construct an optimal clustering for competitive learning. The network automatically constructs the number of competetive and cooperational clusters necessary to represent the structure of the input. Also, the size of each cluster is dependent on the input.

Fourth, the connections are asymmetric. With asymmetrical interaction, population v_i can, in principle, excite v_j even though v_j inhibits v_i, but by using (12), we know that the connections between two populations have the same sign in

both directions. Unfortunately, this makes the system hard to analyze. We cannot easily define a performance index like the harmony function used by Smolensky (1988) or the energy function used by Hopfield (1982, 1984) and Hinton and Sejnowski (1985, 1988).

Formulas similar to (12)-(14) have been frequently used in information theory as measure of transferred information (cf. Hintikka, 1968). We can interpret equation (12) as the information that is transferred from the state of x_i about the state of x_j. This interpretation makes the meaning of the four properties above intuitively sensible.

E. Convergence of Trajectories

In the special case when the weights of the connections are symmetrical, Cohen and Grossberg (1983) have found a Liapunov function that can be used to analyse the behaviour of the network. However, little success has been made in the general case. The Liapunov function has been used to analyse some networks with asymmetrical connections, but it is unclear if that type of analysis can be extended to cover the network presented here. What we would like to do is to find a simple way to predict the path through its state space, the vector $x = (x_0, x_1,..., x_n)$ will follow for every initial value. Especially important is the question of convergence. Do the trajectories of x converge for all initial value of x and if so, to what point?

Computer simulations show that for weights constructed according to (12) the trajectories of x always converge, but the theoretical work that guarantees that this is true in general is still lacking.

3.3 STATIC SCHEMATA

To understand the capabilities of the system characterized in the last few sections we need an interpretation of the neural activity at a higher level. Such an interpretation is developed in this section and shown to constitute a good candidate for a schema level description of neural mechanisms. We will first consider static schemata. A static schema corresponds to a stable pattern of activity in a network. Such schemata will be used in this section to introduce the schema concept and will later be generalized to dynamic schemata which can handle more complex structures.

A. Representational properties and protoypes

Each population in $F^{(a)}$ receives excitatory signals from $F^{(b)}$. As a population responds only to a subset of the activity patterns of $F^{(b)}$, it is appropriate to consider the activity pattern over the populations of $F^{(a)}$ as a categorization of the activity in $F^{(b)}$. With each population of $F^{(a)}$, we can associate a certain *representational property*. There exists a set of activity patterns, P_i, in $F^{(b)}$ for which a population, v_i, in $F^{(a)}$, reacts above a certain threshold. The property associated with v_i corresponds to the activity pattern in the part of $F^{(b)}$ that is shared by all activity patterns in P_i. We may call this common pattern the *prototype* for the property p_i. The activity of v_i, can be considered the strength of the hypothesis that a pattern in $F^{(b)}$ has the property p_i.

B. Prototypicality

It is interesting to note that the activity of v_i is invariant over a large set of patterns in $F^{(b)}$, namely those for which the hypothesis p_i is equally strong. Depending on the activity of v_i for each such invariant set, its members can be placed at a certain distance from the prototype. The closer to the prototype a pattern in $F^{(b)}$ is, the greater effect on $F^{(a)}$. The higher activity in $F^{(a)}$ for more prototypical patterns implies that they are more easily recognized and have greater influence on memory than patterns that are less prototypical. Prototypes are further discussed in relation to the computer simulations in section 4.1.

C. Perceptual properties

In the special case when $F^{(b)}$ is a part of the nervous system involved in perception, the activity of v_i should be understood as a hypothesis on a property of the external world. This interpretation is valid if we take into account what seems to be a general principle of neural representation, namely that of representation by place (Carpenter, 1984, Lansner, 1986). This postulate says that a neural population is always responsible for the representation of the same representational property. Such a property is on a rather low level and does not correspond directly to what is usually understood as a property. Ordinary high level properties such as the shape of an object should be identified with sets of representational properties.

D. Competition and cooperation

The activity patterns of $F^{(b)}$ to which the response of v_i is greater than a certain amount, τ, defines a convex set in the statespace of $F^{(b)}$, namely a sphere of large dimensionality. When cooperation is learned among nodes, the spaces

collapse to their disjunction. If competition instead is learned and the prototypes are sufficiently close, the spaces are changed to the Voronoi tessellation corresponding to the prototypes (Kohonen, 1988). This is the result of the competition among the $F^{(a)}$ nodes. It forces $F^{(a)}$ to choose the property with the largest activity.

E. Definition of a schema

We can now define a schema, α, as an activity pattern in $F^{(a)}$,

$$\alpha = < x_0, x_1, \ldots, x_n >, \quad (15)$$

where x_i is the activity of population v_i and n is the number of populations in $F^{(a)}$. Since we equate each population in $F^{(a)}$ with a representational property p_i, we may also consider a schema as a set of such representational properties, $\{p_i, p_j, ..., p_k\}$, where p_i is in the set if x_i is greater than the threshold φ. In the rest of this paper, these two interpretations of a schema will be used interchangingly.

The schemata self-organize according to the system equations (5)-(14) defined above. After some experience with the environment, in this case the patterns in $F^{(b)}$, the system will construct schemata that represent different aspects of the environment. Let us consider some of the different relations that can exist between two schemata, α and β.

F. Schema containment

Let $\alpha = <\alpha_0, \alpha_1, \ldots, \alpha_n>$ and $\beta = <\beta_0, \beta_1, \ldots, \beta_n>$. If for each i $\alpha_i \leq \beta_i$, then we say that α is *contained* in β. The schema α can be considered a more general schema than β. The schema β is an instantiation of the schema α. The part of β not in α, is a variable part of the schema α. This implies that all schemata with α as a subset can be considered to be an instantiation of α with different variable instantiations. Thus, schemata can have variables even though they do not have any explicit representation of variables. Only the value of the variable is represented and not the variable as such. If we consider schemata sets of representational properties, the notion of containment corresponds to set inclusion.

The index of the instantiation is identified with the added properties in β compared to α. The possible instantiations are constrained by the weights of the connections in the low level neural system. This is the mechanism responsible for conditions on variable instantiations and default values. Since we can speak of a containment relation on schemata, it follows that there is an elementary possibility for the embedding of schemata in each other. However, for the tasks usually associated with schema processing, this elementary form of embedding is not sufficient. For example, self embedding is not possible. In fact, the possible embeddings are even more limited if we accept definition (15) of schema above. Since each property can only occur once in the schema, only schemata that do not share any properties can be embedded. This is truly too limited and some added mechanism is needed. Such a mechanism will be developed in section 3.5-3.7.

G. Generalization of schemata

Define the generalization of two schemata α and β as a new schema γ, where for each i, $\gamma_i = min(\alpha_i, \beta_i)$. In the set interpretation of schemata the generalization corresponds to the disjunction of the two sets α and β. If this disjunction, γ, is not empty, the two schemata are similar. For the generalization of two schemata α and β, we will use the notation $\alpha \oplus \beta$. The larger the generalization is, the larger the similarity. This can be the starting point of a definition of a similarity measure for schemata. If α and β are disjoint sets, they are not at all similar. This simple idea of similarity is only justified if the projection from $F^{(b)}$ to $F^{(a)}$ preserves the sizes of the patterns. This does not hold in general, but since learning in the $F^{(b)} \rightarrow F^{(a)}$ projection will not be discussed in this paper, the definition above is sufficient for our present purposes. Since each property is physically represented in a specific place, two similar schemata cannot be represented simultaneously without being mixed. This may indeed sound limiting at first, but is only, if we want everything to be actively represented at the same time. This is not at all necessary and the advantages of using a dynamic representation that changes over time will be discussed below.

H. The coincidence of schemata

Define the coincidence of two schemata α and β as a new schema γ, where for each *i*, $\gamma_i = max(\alpha_i, \beta_i)$. If we interpret schemata as set of representational properties, the coincidence of two schemata is the schema that contains all the representational properties of both the original schemata. In other words, coincidence corresponds to the union of sets. For the concidence of α and and β, we write $\alpha \bullet \beta$.

I. Implicit associations

Define $Imp(\alpha, \beta)$ as the sum of the signals sent to schema β if schema α is active. Thus, $Imp()$ is a function from two schemata to $\Re$. The strength

of this function for all different α and β determines the model of the environment built by the network. This is the schema level version of connective weights between cell populations. Depending on various assumptions on the neural level, the definition of *Imp*() will be looking different. No explicit example will be worked out here. This is called an implicit association since the association is not explicitly represented in the schemata, but rather by the connections in the network. Note that while the generalization and coincidence are operators on the space of schemata, the *Imp*() function is of a different type.

J. Resonant schemata

From the *Imp* function it is a straight forward matter to define a function from each schema α to another schema $[\alpha]$. $[\alpha]$ is the schema that get activated as a result of the initial activation of α. This is called the resonant schema corresponding to α. It corresponds to the equilibrium state that the neural level approaches with a certain input corresponding to the schema α. We can also easily define an equivalence class to each schema α. This is the class of schemata for which the resonant schemata are equal to $[\alpha]$.

K. Explicit association

Define $Exp(\alpha, \beta)$ as a monotone increasing function of the size of the generalization of schema α and β. The exact shape of the function depends on assumptions on the neural level. *Exp*() is a similarity measure for schemata. It can also be used as a starting point for a discussion of associations between schemata. The larger Exp is the larger the association. Since the similarity is explicitly represented in the schema, this type of association is called an explicit association. How such associations are used to form links between schemata will be discussed below in section 3.4.

L. Categorisation and Association

The projection from $F^{(b)}$ to $F^{(a)}$ allows $F^{(a)}$ to act as a set of categorizers for the patterns in $F^{(b)}$. The neuronal field $F^{(a)}$ acts as an auto-associator whose items are the categories formed in the $F^{(b)} \rightarrow F^{(a)}$ projection. The two processes of categorization and association are responsible for all processing in the network described above. It is possible to use the schema formulation to show, in a formal way, how these two processes are related. Such an analysis has been made in Balkenius (1992). There is a simple intuitive relation between the process of categorization and the in-star architecture and association and the out-star architecture. The out-star can be said to read out an association to the network (figure 2).

3.4 INTERMEDIATE MEMORY AS A BASIS FOR MULTIPLE INSTANTIATIONS, LINKS AND EMBEDDINGS.

Definition (15) above excludes the possibility of the representation of similar (in the sense defined above) schemata simultaneously. The main advantage of this kind of representation is that everything the system learns about a schema is automatically generalized to all instances of that schema. Since no slots are used, there is no need to move information from one slot to another. The motivation for this type of schema representations comes from the decision to avoid strong assumptions about the number of slots used for representations. It is also unclear how slots could self-organize in a biological neural network if they are not explicitly present in the input.

To represent more than one schema, a mechanism for intermediate memory is introduced. It is important to realize that even if we postulate a certain number of slots for representing different items, we need a mechanism for saving the old contents of those slots when their contents are changed. For instance, consider the case of linguistic representations. Assume that we have one slot for each semantic role in a sentence. This works well if we want to represent only one sentence, but as soon as we want our schema system to work with more than one sentence at a time, we need some kind of storage mechanism. We can as well introduce such a mechanism at once and leave the problems with slots behind at the same time.

To do this we need to do one additional assumption at the neural level. The effect of the activity in the presynaptic population on the postsynaptic one is assumed to be temporarily raised after simultaneous activation of both the pre- and postsynaptic population. This can be the result of one of the following neural mechanisms.

A. Temporary lowering of the threshold in a cell group, v_{ij}, after activation.

B. Temporary sensitation of the cell group, v_{ij}, after activation.

C. Temporary depolarization of the synaptic junctions after activation.

Each of these three mechanisms above can be approximated by,

$$w_{ij} \approx \max(0, \mathbf{F}(p(j|i)-p(j); 1-p(j))) + IM_{ij} \quad (12')$$

where IM_{ij} is an intermediate memory term and,

$$\mathbf{F}(d;dt)IM_{ij} = -cIM_{ij} + \delta x_{ij}, \quad (16)$$

or alternatively,

$$\mathbf{F}(d;dt)IM_{ij} = -cIM_{ij} + \delta x_{ij}x_{ji} \quad (16')$$

This will function as an intermediate memory by temporarily changing the set of resonant states. Thus, the resonance function depends on the *IM* matrix (a similar view of temporary changes of neural connectivity can be found in von der Malsburg and Bienenstock, 1986).

Example 1: Intermediate Memory

This example tries to explain the effect of the added mechanism. Suppose we present two schemata, α and β, to the system by manipulating the activity in $F^{(b)}$. Assume that $[\alpha]=\alpha$ and $[\beta]=\beta$. Also, suppose that the generalization of α and β is γ, i. e. $\gamma=\alpha\oplus\beta$, and that there exists a subset of α, denoted by α', that is not in β. First, present the schema β to the network then reset the activity of $F^{(a)}$ (figure 3a and b). If we now present the schema γ to the system, the schema β will be recalled (figure 3c). This is because the intermediate memory have changed γ to β, i.e. $[\gamma]=\beta$. Thus, the most salient schema that is similar to γ is recalled. If, instead, we would have used α' as input, the schema $\alpha=[\alpha']$ would have been recalled (figure 3d). This mechanism works for a large number of simultaneously represented schemata in an intermediate memory. The index of each instantiation of a schema is identified with the part of the schema that distinguishes it from other similar schemata. The limiting factors are the similarity of the different schemata in *IM*. The more similar they are, the larger the probability that they will be mixed with each other.

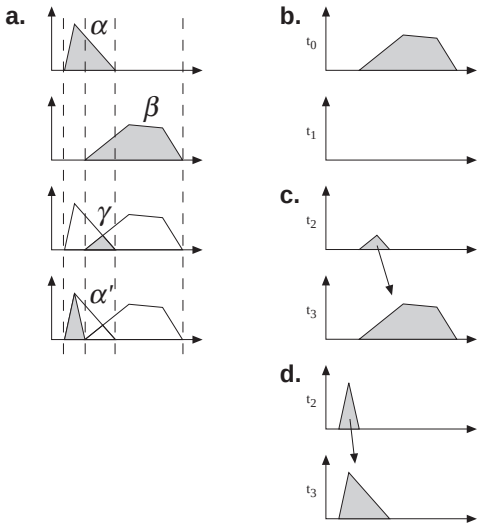

Figure 3. (a) The different schemata; (b) β is stored in intermediate memory; (c) γ is used as a cue to recall β; (d) α' is used as a cue to recall α.

Example 2: Explicit linking

The intermediate memory can be used to represent schemata that are linked to each other by means of an explicit association relation. Let us say that we want to represent a relation, ρ, between two schemata α and β. This is accomplished by storing two patterns in the intermedient memory. First, the coincidence of α and ρ are stored in *IM*, and then the coincidence of β and ρ (Figure 4a and b). This associates the two schemata α and β by means of the undirected relation ρ. If we want the relation (or link) to be directed, all we need to do is to add to each instance of ρ some information referring to the direction of the relation. Let ξ represent the schema for being first argument to the relation and ψ the schema for being the last. The directed relation is represented in *IM* as two schemata, namely, $\alpha\bullet\xi\bullet\rho$ and $\beta\bullet\psi\bullet\rho$ (Figure 4c). To recall each part of the relation, we present only a part of the relation, say $\xi\bullet\rho$ to recall $\alpha\bullet\xi\bullet\rho$ or possibly a smaller part, if only a few schemata are stored in *IM* (Figure 4d). This mechanism works for any number of arguments to a relation. It should be noted that this mechanism also can handle some forms of self-embedding. It is important to realize that none of the schemata α, β, ρ, ξ and ψ above necessarily must be exactly the same each time they are represented as long as they are sufficiently similar. This implies that schemata can have a radial structure without violating the mechanisms described here.

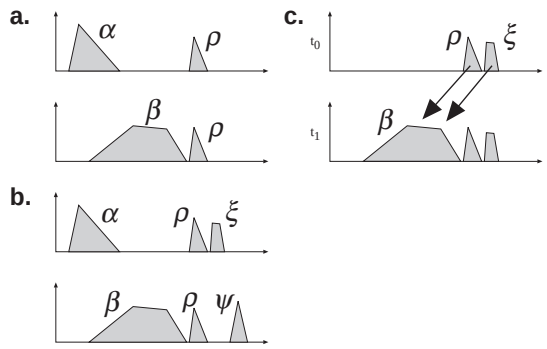

Figure 4. Explicit linking. The figures are explained in the text.

This completes the description of the system for static schemata. The rest of the paper develops some possible neural architectures for systems capable of dynamic schema processing. Computer simulations of the static schema system can be found in section 4.1-4.4.

3.5 REPRESENTING SEQUENTIAL STRUCTURE

A. Mapping time into space

The usual approach to representing sequential information in a neural network incorporates the idea of mapping time into space. The most commonly used method is to represent the input each time at a different place. Using this idea we need a different physical slot for each time slot we want to be represented. This implies that the number of slots used must equal the number of events that we are going to represent in the network. This is unsatisfactory for several reasons. First, it shares the problems discussed above with other mechanisms based on slots. Second, as time moves on, we need to sequence the inputs through the slots by means of a shift operation. How this could be realized by a real neural network is not clear for those models that incorporate such mechanisms. It is highly unlikely that a sequential buffer can be seen as a biologically plausible mechanism (Grossberg 1978, 1986).

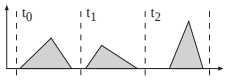

Figure 5. Time slots. Different parts of the network are used to represent the schemata at different times.

B. Ignoring temporal structure

Another approach used by some researchers is to ignore the dynamic aspects of the input totally, and represent it directly as a spatial pattern thereby mapping the temporal problem onto a spatial one. (e.g. Rumelhart and McClelland, 1986) Since the mechanisms for static representations are much better understood, this may be a tempting route to follow, but it must be taken into account that when the dynamical problem is mapped onto a static one, much of the original problems are lost, and thus, the explanative power of the static model is reduced.

C. Intermediate memory as virtual slots

If we accept definition (15), neither of the above mechanisms are acceptable. Fortunately, the intermediate memory mechanism suggests a solution. If we present a sequence of schemata to the system, then each of them get stored in *IM*, which acts as a set of virtual slots. Each slot can be accessed by activating a sufficiently large part of its content. This representation overcomes the problem of choosing a fixed number of slots. A sequential input of schemata is stored as a set of potential schemata that are available for recall from *IM* at any time.

What is missing now, is a representation of the sequential ordering of schemata that can be recognized by another part of the system. The introduction of such a representation is needed to process temporal structure. Consider a neural circuit, G, with the following properties:

A. Each population in G receives input from one population in a constraint satisfying network ($F^{(a)}$) that is capable of static schema processing and intermediate memory.

B. As a population in $F^{(a)}$ increases its activity the corresponding population in G receives a short pulse of activity. This can be generated by a number of neural GATE mechanisms (figure 6). The pulse causes the state of a sequence sensitive DECAY module (figure 9) to change in one of two ways. The first type of change causes the network to be order sensitive while the second type makes it order and rhythm sensitive.

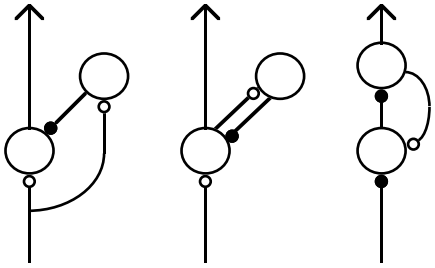

Figure 6. Three GATE mechanisms. As the input to the lower population turns active, a short pulse is emitted from the circuit.

D. An order sensitive circuit

The pulse increases the activity of the population receiving it and decreases the activation in all other populations either as a result of on-center off-surround projection from G to S (figure 9a) or as a result of lateral interaction within S (figure 9b). These mechanisms are said to support active decay of the activity in S since the activity decreases only when S receives an input. Active decay makes the representation in S independent of the time between the impulses to $F^{(a)}$ and thus makes S sensitive to order but not to rhythm (figure 7).

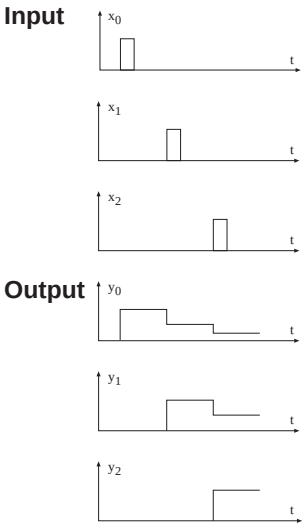

Figure 7. The representation of order genrated by the network in figure 9a. The input to the circuit is denoted by x_i and the output by y_i.

E. A rhythm sensitive circuit

The pulse from G increases the activity of receiving population but leaves all other activities in S unaffected. Instead they decay passively. This mechanism is said to support passive decay and makes S both order and rhythm sensitive (figure 8 and 9c).

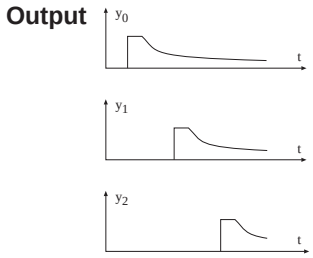

Figure 8. The representation of order and rhythm generated in the circuit in figure 9c. The input is the same as in figure 7.

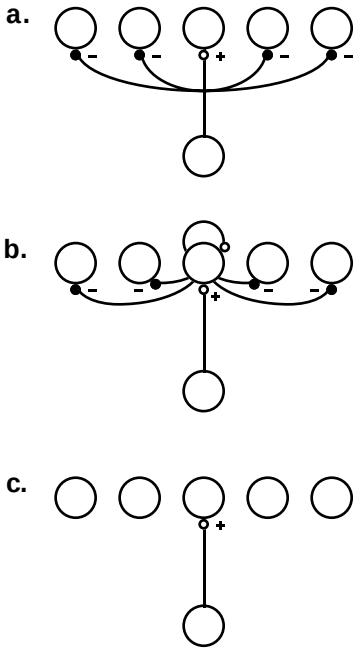

Figure 9. Three DECAY mechanisms. (a): Active decay as a result of signal read-in. (b): Decay due to normalizing properties within the field. (c): Passive decay within each cell population.

The effect of each of the mechanisms above is to leave a STM trace in S that encodes the sequential ordering of the input to $F^{(a)}$. The more recently the schema is received by $F^{(a)}$ the stronger the activation of the corresponding populations in S. Thus, these mechanisms produce an recency gradient in S. (cf. Cohen and Grossberg, 1986, Grossberg, 1973, 1986) The more recent an event, the more active is its representation. This is a mechanism that maps time into space and is consistent with (15).

F. Multiplicative decay

A particularly interesting case occurs when S supports a multiplicative decay, i.e. the changes of the temporal representations can be written as,

$$x_i(t+\Delta t) = c^{\Delta t} * x_i(t), \qquad (17)$$

where if x_i is the activity of population i in S and c does not depend on i. This leaves the relative activities of the different population constant even if the input to S is increased (see figure 8 and 10).

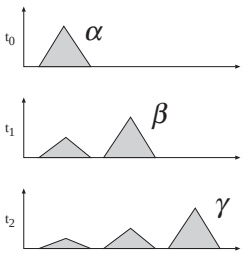

Figure 10. The activity of the three populations in $S^{(1)}$ as a result of presenting the schema sequence (α, β, γ) to the system.

3.6 RECOGNIZING AND CLASSIFYING SEQUENTIAL STRUCTURE

The last section describes how a sequence of schemata could be converted to a spatial code. The system has now to recognize these different sequential orderings that are represented in S. This can be done in a number of ways. The simplest mechanism, capable of self-organization of recognition classes, is a network using simple competitive learning (Rumelhart and Zipser, 1988). But, as discussed by Grossberg (1980), simple competitive learning cannot form a stable categorization of arbitrary input vectors. The problem is especially apparent when the input vectors are highly linearly dependent as is the case with the activity vectors in S. A number of different options are available for the solution to this problem.

A. Masking fields

One solution is to use a 'Masking Field' architecture (MF) for recognition of the activity vectors in S (Grossberg, 1986, 1987). MF:s are especially interesting because of their ability to segment input into substrings with lengths that are highly context sensitive. The main idea is to exploit a multiplicative decay property in S and classify all subsequences represented in S simultaneously. Then the different classifications of the subsequences are allowed to compete with each other to form a stable segmentation. The architecture of a MF is highly complex and it would lead too far to describe it in any detail here (for an extensive analysis refer to Grossberg, 1986 or Cohen and Grossberg, 1986, 1987).

B. Self-organizing topology preserving maps

A simpler mechanism is the self-organizing topology preserving mapping presented by Kohonen (1988). Here, a set of activity vectors of large dimensionality can be mapped onto one single, or a few, dimensions. The mapping preserves topology by letting neurons that are close to each other code similar activity patterns. Although the coding is not stable, recoding can only be performed as a continuous movement of parts of the map. This makes the problem of instability smaller as long as the recoding process does not proceed to fast. The mechanism for producing a mapping from S to the recognition field R can be characterized in the following way (figure 11):

Competition The populations in R compete for the coding of an activity vector in S. The population that receives the largest input is selected as the winner.

Cooperation The winning population excites populations physically close to it as a decreasing function of distance.

Learning The winning populations and its neighbors are allowed to change their receptive fields towards the activity vector of S to make them more responsive to the input pattern.

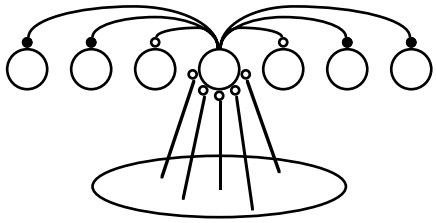

Figure 11. Laterally interconnected populations capable of forming a topology preserving mapping from the activity patterns in the lower field to the upper.

This classification process is capable of constructing a taxonomic map of the activity vectors in S that are distributed over the nodes of R.

With one of the afore-mentioned mechanisms, it is possible for a schema processing system to construct a classification for the different

sequential orderings of static schemata that appear in $F^{(a)}$. This classification can be used as a basis for the prediction of the next schema in the sequence, or as a trigger signal for sequential production of schemata in $F^{(a)}$.

3.7 PREDICTION AND PRODUCTION

A. Sequential read-out

The activity patterns in S show a recency gradient with respect to the temporal ordering of the schemata in $F^{(a)}$. There exists a number of neural mechanisms that take a pattern with a primacy gradient as input and produce a temporal sequence of events as a result. Such a mechanism does the opposite of G. While G takes a sequence of events as input and produces a recency gradient, our next mechanism takes a primacy gradient as input and reads out a temporally ordered sequence of events. For this, we need two modules, R-I and R-O, for sequential read-in and read-out (figure 12). We also need two levels of sequence representations $G^{(1)} \rightarrow S^{(1)} \rightarrow R^{(1)}$ and $G^{(2)} \rightarrow S^{(2)}$. Let us first consider how a read-out mechanism works and then proceed to how the primacy gradient is constructed.

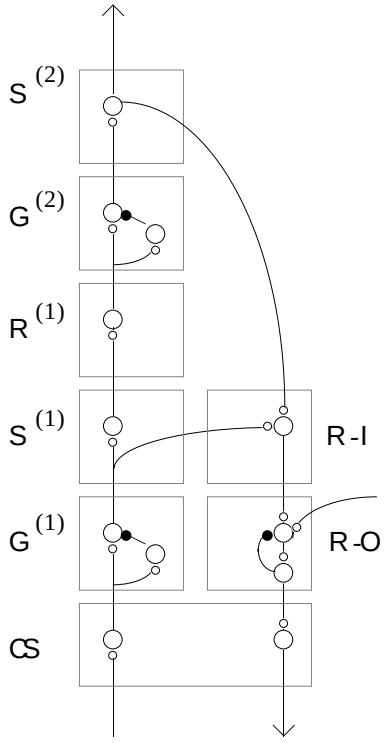

Figure 12. The interaction between the different neural modules in the schema processing system. CS: constraint satisfying net ($F^{(a)}$ and $F^{(b)}$); $G^{(1)}$ and $G^{(2)}$: gate circuits; $S^{(1)}$ and $S^{(2)}$: sequential representation; R-I:sequential read-in module; R-O: sequential read-out module.

Assume that the larger the signal in the connection from a population in S2 to R-I, the faster a signal is sent from R-I to R-O. Now suppose that the mapping from $S^{(1)}$ to R-O represents a primacy gradient. If a single population in $G^{(2)}$ is activated, it will in turn activate a sequence of populations in R-O. Let activation of a population in R-O emit a short pulse to its corresponding population in $F^{(a)}$. This could be accomplished by means of an inhibitory connection from each node in R-O to itself, or through an interneuron. This mechanism is appropriate for the read-out of the primacy gradient coded in the mapping from $S^{(2)}$ to R-I into $F^{(a)}$. The read-out mechanism may be modulated by a read-out signal that non-specifically activates all the nodes in R-O. This is approximately the approach of Grossberg (1986) and Cohen and Grossberg (1986). Another read-out mechanism that uses a primacy gradient to read-out sequences is discussed by Crick in his model of attention (1984).

B. Learning of a primacy gradient

The last problem to consider is how the primacy gradient is learned in the mapping from $S^{(2)}$ to R-I. Let R-I receive inputs from $G^{(1)}$ or $F^{(a)}$ in such a way that each time a population is activated in $F^{(a)}$, the corresponding population in R-I is activated for a short time. If the activity of each population in $S^{(1)}$ is used as a sampling signal for the corresponding population in R-I, the amount of the pattern in R-I that $S^{(2)}$ samples depends on how recently a certain sequence was produced in $F^{(a)}$. This produces a primacy gradient in the mapping from $S^{(2)}$ to R-I. The prediction read-out at each time can be approximated by,

$$PRED_t = SS_t \backslash \mathbf{I} \backslash su(u=to;t;\ SS_u\ RI_u\ ), \qquad (18)$$

where SS_t is the sampling signal at time t, and RI_t is the read-in signal at the same time.

C. Predicting a schema sequence

To predict schema sequences, we need to expand the system with two modules with capabilities similar to $G^{(1)}$ and $S^{(1)}$. These modules, $G^{(2)}$ and $S^{(2)}$, represent the temporal ordering of the classifications in $R^{(1)}$. The greater the activity in a population in $S^{(2)}$, the more recently the corresponding sequence was recognized by $R^{(1)}$.

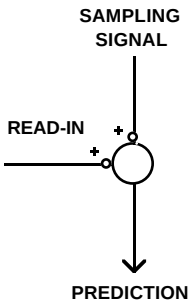

Figure 13. Sequential Read-In module (R-I). The sampling signal determines the amount of read-in signal sampled by the population. The prediction is approximately equal to the product of the sampling signal and the average amount of sampled read-in.

Example 1: Learning a schema sequence

Assume that the sequence of schemata (α, β, γ) was produced in $F^{(a)}$. When α is produced, the sequence (α), (of one item) is represented in $S^{(1)}$. It is now possible for $R^{(1)}$ to recognize that sequence. This in turn causes $S^{(2)}$ to represent that the sequence (α) was recently recognized. At the same time R-I is activated with α, it receives a sampling signal from $S^{(2)}$ that lets the $S^{(2)} \rightarrow$ R-I connections sample the pattern in R-I. Now β is presented to the system. This causes $S^{(1)}$ to represent the sequence (α, β) which in turn is recognized by $R^{(1)}$. This activates a new population in $S^{(2)}$ that simultaneously decreases the representation for (α). This causes the sampling signal from $S^{(2)}$ to R-I to decrease, and the second schema β is sampled with less intensity than α. At the same time the newly activated population in $S^{(2)}$ lets a different population in R-I sample β with full intensity, etc. The result is that the sequence (α) produces the prediction of (β, γ) in R-I on later trails. Note that it will only take one learning trail to learn a sequence with this mechanism. It is also possible for the system to work on several sequences simultaneously if $R^{(1)}$ use a MF-architecture. Another possibility is to use several layers of dynamical systems to handle hierarchical structures in the input.

Example 2: Syntactic generalization

An interesting case occurs when we feed sequences of similar schemata to the system. Say that, we repeatedly feed the system the three sequences, (α, β, γ), (δ, ε, ϕ) and (σ, τ, υ) where the first schema in every sequence is similar to every other first schema, i.e. they have some common subschema. The same is true for the other two schemata in the sequence. Call this common subschema sequence (ξ, ψ, ζ). An input of, say, (α), make the system predict the schema sequence (β, γ)) as the next two schemata. It also predicts the sequence (ψ, ζ) since they are subschemata of β and γ. Suppose that *IM* contains the three schemata α, ε and υ, and α is actively represented in STM. If the system now receives a read-out signal, it will read out the sequence (ψ, ζ) into $F^{(a)}$. This causes $F^{(a)}$ to represent the two schemata [ψ] and [ζ] after each other. Let us assume that due to the contents of *IM*, [ψ]=ε and [ζ]=υ. The actual sequence produced in $F^{(a)}$ will be (α, ε, υ). This is a straightforward generalization from the observed sequences. Of course the schema sequence (α, β, γ) will also be produced but the contents of *IM* forces it to be ignored by $F^{(a)}$. Whether this is the case or not depends on the relative importance of the different schemata and the past experience. It could also happen that (α, β, γ) would be produced in $F^{(a)}$ if the schemata ξ, ψ and ζ were very small relative to α, β and γ, but in general the more abstract sequence would be favoured because it appears more frequently in the input. If we consider the example again, what have been learned by the system is a preferred sequential ordering of the more abstract classes (ξ, ψ, ζ) and not only the individual sequences. With the experience of the three sequences, the system could predict the class membership of a new schemata, Ψ, in a new sequence such as (α, β, Ψ). This could be the mechanism used to form hypotheses about the semantic or syntactic class of a word that has never before been presented to the system.

4. COMPUTER SIMULATIONS

The rest of this paper describes a number of computer simulations performed with the system described above. The first four investigates the different properties of the constraint satisfying module ($F^{(a)} \leftrightarrow F^{(b)}$). The others demonstrate the capabilities of the dynamic system. It is important to note that all the computer simulations used exactly the same architecture. It is also of interest to notice that the network uses unsupervised learning. For the simulations described below, this means that the system did not receive any information about what it was supposed to do. Neither did it receive information about whether it was in the learning phase or in the test phase. Thus, all the abilities demonstrated below self-organizes from observations in the environment.

4.1 PROTOTYPE EXTRACTION

A common experiment using neural networks investigates their ability to find prototypes in a data set. Since the $F^{(a)}$ module of the system is constructed mainly for the construction of associations between schemata and not for classification, this task is especially interesting to investigate. Two kinds of simulations were performed.

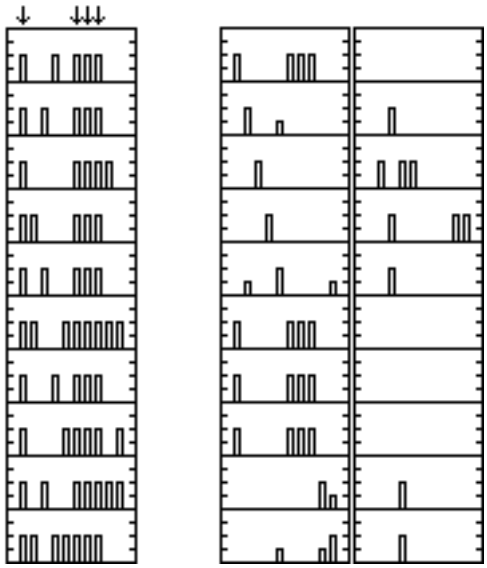

Figure 14. The result of presenting one schema with added noise to the system. The left figure shows the 10 inputs that were given to the system. The protoype bit and the 3 bits that constitutes the schema are marked. The right figure shows the receptive fields for each of the nodes after learning. The first row shows the receptive field of the schema bit. This is exactly the schema in the input data. The left part of the diagram describes the excitatory connections and the right part the inhibitory connections.

A. Noise detection

In the first simulation, the system was given a set of prototypes as input. The inputs were represented as orthogonal binary vectors. Noise was added to each instance of a schema. The noise was uniformly distributed over all inputs and consisted of bits of the vector turned from 0 to 1. To check the learning, a prototype bit was associated with each prototype. The prototypes found in the data set could be recognized by looking at the receptive fields corresponding to each prototype bit. The prototype bit does not influence the result of the learning but makes it easy to recognize the prototypes if they are found in the data. The result of the simulation showed that all prototypes were found in the data set independent of the sizes of the schemata and the network. Thus, the noise was treated as noise and did not influence learning. This result is an obvious consequence of the learning equations that govern learning in $F^{(a)}$. A scaled down version of the simulation can be found in figure 14.

B. Prototypes with variables

The distribution of the noise in the second simulation depended on the prototype in such a way that it could be considered to be different variable instantiations of the prototypical schema. This means that each prototype had its own noise that occurred together with that prototype only. The alteration of the noise changed the prototypes learned by the prototype bits to include the prototype together with the noise. Again, this is a consequence of the learning equations for $F^{(a)}$. The learned prototype is such that each instance of the schema has a large influence on the prototype bit (figure 15).

4.2 SCHEMA RESTORATION

Schema restoration is a task in which the system is given a schema as input and has to figure out if some part of it is missing or should not be there. Simulations were performed on a number of network sizes and for different number of schemata of different sizes. The schema prototypes learned by the system were entirely randomly generated. Noise was added to the inputs at the test trails by switching bits from 1 to 0 or from 0 to 1. The simulation showed that even with a reasonable small number of prototypes with sizes that were approximately an eighth of the network size almost no schemata were restored correctly with as little as 3% noise. The reasons for this bad behaviour have to do with the generation of schemata and the ideas behind the learning equations. Since the prototypical schemata are randomly generated there is almost no structure to be found. This makes the coding of constraints hard. A second problem relates to the learning equations. As a result of them, a bit that is changed from 0 to 1 is treated differently from a bit that is changed from 1 to 0. In the first case, pieces of added information is given to the system. The single bit can be in conflict with the rest of the input. In most cases, it certainly is because it did not belong to the input schema. This single bit will strongly inhibit the rest of the input and move the system away from the correct prototype. In the second case, when a bit is missing in the input it can be easily corrected by the rest of the input. No interference occurs since a missing bit does not cause any activity in the system. As a result,

we would expect the system to be bad at restoration but very good at associating to missing information. This is studied in the next simulation.

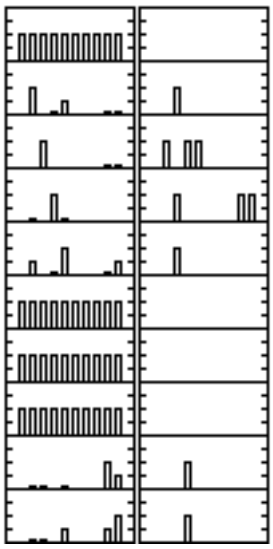

Figure 15. The resulting receptive fields after exposition to one schema with varying variable instantiations. The prototype bit responds to all possible instantiations of the schema.

4.3 SCHEMA ASSOCIATION - NAMING

Schema association is a task in which the system learns a set of pairs of schemata. On later trails the network has to recall both schemata in the pair when one of them is given as input. Two kinds of simulations were done: random associations and structured associations.

In the first kind of simulation, the system was taught random schemata of a certain size (SS). In the test phase, parts of the schemata previously learned were presented. The task for the network was to recall the missing parts of the presented schemata. Simulations were performed with different network sizes (N) and with different number of schemata (NS). The amount of the schemata that were missing was also varied (PD). The system performed this task fairly well. Even when as much as 60% of the schemata were missing, 79% of the them were recalled correctly, in average (N=100, NS=32, SS=16). When the schemata were smaller compared to the network size, the result was even better.

The second simulation was a so called naming or identification test. Two sets of schemata were constructed. The first, NAME, contained the schemata for different names. They were represented as a set of orthogonal vectors. The second set, THINGS, contained a number of schemata for things that corresponded to the names in NAME. The things were represented as random binary vectors. Tests were made after different number of expositions to the pairs of schemata. The performance proved to be excellent. The associations in both directions between NAME and THINGS were correct 100% of the time after as little as one exposition to each pair in the training data. An interesting variation of the test is when the input pairs are not exactly the same every time they are presented. Two kinds of such deviations from the prototypical pairs were introduced.

A. Differentiated schemata for things

The first kind of deviation distorts the schemata for things a little on each learning trail. This corresponds to observations of a set of things that are members of the same class and called the same name. This transforms the naming task to a prototype extraction task (simulation 4.1).

B. Wrong name

The second error that can be introduced is to use the wrong name for things in some of the expositions. Again, the performance of the system is very good. The number of errors made depends on how often the wrong name is paired with a certain thing. If the erroneous names stands for things with schemata similar to the presented schemata, more expositions to the correct pair is necessary before the error is corrected. But the system will forget the erroneous namings in the long run and start to use the correct name for a thing. How systematic the errors are is also important. If they are completely random, almost no effect can be found on the system if many expositions are performed.

4.4 CONSTRAINT SATISFACTION

The static schema system was constructed to perform constraint satisfaction and we would expect it to behave very well on such task. This is indeed the case and can be seen from the result of this last simulation with the static system. The task was similar to the naming task described above. The difference was that a simple environment was simulated for the system to observe.

The environment consists of four persons (John, Sam, Ann and Mary) that can perform four activities (Kick, Hit, Kiss and Love). Each of the persons can be either the agent or the patient of

each of these activities. Each person is represented as a binary vector with a certain bit for each person and some bits (4) that are similar to all persons. This makes it possible for the system to recognize both similarities and differences among people. The coding of the different entities in the world can be seen in figure 16.

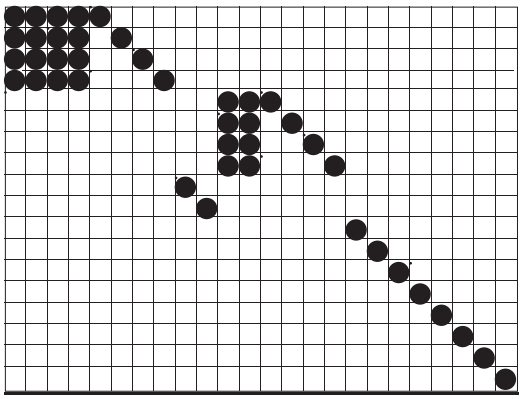

Figure 16. The representation of the environment. Each row represents an item in the environment. A filled area represents an activity value of 1 while an empty area corresponds to 0. The 4 first are the four persons, the 4 next are the activities, the next 2 represent whether a person is active or not, and the eight last represent the eight words used. What the individual dots represent does not matter as long as they contain enough information, i.e. similar items have similar codings and distinc item have distinc coding.

The result of the simulation is show in figure 11. A rich structure has been learned. This structure completely describes the observed environment. The schemata that can occur in the system are exactly those of the observed environment. This demonstrates the ability of $F^{(a)}$ to form constraints between the different schemata. When the structure of the environment can be described by such constraints, the storage capacity of the system is very high. The theoretical limit is 2^N, if N is the network size. This is of course an environment were exactly everything occurs. All combinations cannot be constructed by the sytem, but the number of stored states can be as large as 2^N if the environment has a certain structure.

Figure 18 shows the connections in figure 17 in an alternative way. The different nodes have been sorted into groups according to their distribution in the input patterns. Nodes that excite each other are connected with lines.We can see that a name of a person will come to excite both the general person schema and a specific person schema for that person. The network can also be used the other way around. A person specific schema will activate the general person schema as well as the name schema of that person..

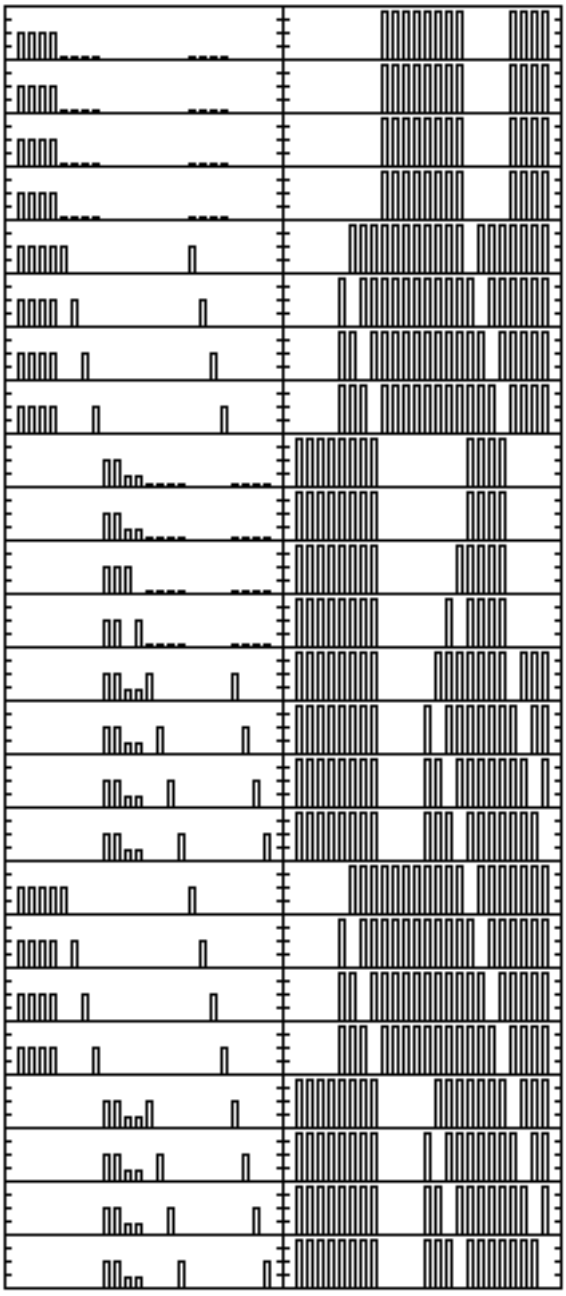

Figure 17. The connections in $F^{(a)}$ after presentation of the environment. To the left in each row are the strengths of the excitatory conections from the corresponding poulation. To the right, the inhibitory connections from each population are shown. If we consider each node an out-star, the pattern shown to the left of each row indicates the pattern that will be read out if we activate the corresponding population.For example, node 8 will excite nodes 1 through 4, node 8 and node 20 while the other nodes will be inhibited.

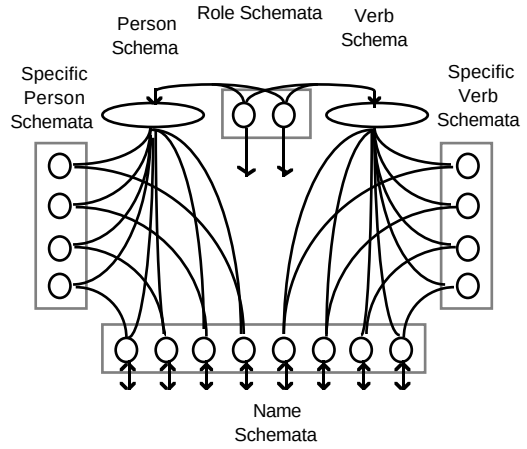

Figure 18. The configuration of $F^{(a)}$ when the structure of the environment is learned. The dotted frames represent competetive clusters among the schemata. Such clusters were formed for name schemata, activity schemata, person schemata and role schemata. This captures the structure of the simulated environment exactly. Lines show cooperational interaction among schemata. They are directed where shown. Each persons schema activates the corresponding name schema. The same is true for verb schemata. Role schemata excites both the person schema and the verb schema since a role was always assigned to a person when he was involved in an action. For the same reason, it also excites all other schema but not as strongly. On the other hand the role nodes does not receive any connections from the person or verb schema since they do not contain any information about the roles. Not only does the network code the structure of the environment but also who is most likely to do what. In the figure shown here, everything was equally likely to happen. If this had not been the case, we would expect connections between the person and the verb schemata nodes.

4.5 TAXONOMIC CLASSIFICATION OF SEQUENCES

The set of word schemata from simulation C and D was given to the system. The word order was varied and the dynamic system was allowed to learn the different orderings. This causes a map of the different orderings to be constructed in $R^{(1)}$. Nodes close to each other code similar word order. The type of mapping is demonstrated by table 1.

4.6 PREDICTION

The dynamic system is capable of recognizing and predicting sequences. The task in D was performed again using 'real' sentences with three words to describe the situations observed in the environment. The learning task made extensive use of intermediate memory to temporarily store samples of the environment.

Node	Word sequence
1	John kisses Mary
2	John kisses Ann
3	John hits Ann
4	John hits Sam
5	Sam hits John
:	:

Table 1. A part of the map formed in $R^{(1)}$ as a result of observing different sentences. Similar sequences are coded by nodes that are close.

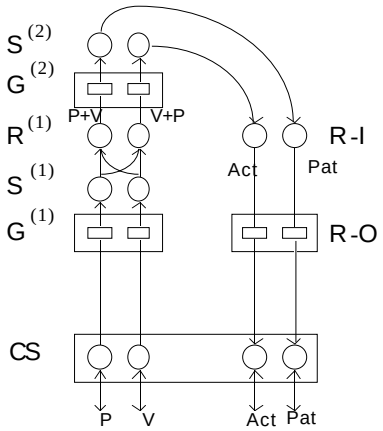

Figure 19. The pathways constructed through the dynamical system as a result of the self-organizing process. Only two pathways are shown. If the person schema is activated first and the verb schema later, $R^{(1)}$ will recognize this sequence and predicting the role agent to the person. If instead the person schema is activated last and the verb schema first the role patient is predicted. This shows that the network has learned some elementary facts about word order.

A typical input was a three word sequence such as "John loves Mary" together with two samples of the environment representing John • Loves • Agent and Mary • Loves • Patient. Note that the sentence input was given in real time and not presented as a single pattern. The dynamic system had to recognize the different ordering of the inputs and learn to assign roles accordingly. When about half of the possible sentences had been shown, the system was tested with word inputs alone. The task for the system was to construct the corresponding semantic information. This was accomplished in all cases. The result of the self-organization in the dynamical system can be seen in figure 13. The word order Person • Verb is associated with the Agent schema and the word order Verb • Person is associated with the Patient schema. This is a very simple example of what can be done with

the system, but it nevertheless shows that real time parsing is possible in a neural network and can be taught from examples. It is obvious that the mechanisms described here work for longer sequences too. Only the size of the dynamical system limits the number of different sequences that can be learned. It is also possible to put a second dynamic system on top of the first in order to recognize hierarchial structures.

4.7 PRODUCTION

Finally, the system from simulation F was given only the environmental information as input together with a production signal. This had the desired effect and the sentences were read out with the correct word order (Table 2).

This example shows that production of sentences can be generated by the same mechanism that perfom the parsing. The content of CS is resposible for the prediction of the sequence. Depending on CS different sentences can be produced. If the desired information is not in CS, it is recalled from *IM* if possible. Simulations have been performed in which the contents of CS had to select either an active or a passive sentence. This too was accomplished very easily. Note that the system did not learn every sentence in advance. The syntax observed could be generalized to new situations in which the system had never received any sentences.

5. CONCLUSION

This paper has shown how it is possible to construct a neural network that is capable of self-organizing the processing of schemata and sequential information. The basic building blocks of this system are the constrain satisfying network which incorporates categorisation, association and learning and the dynamic system capable of recognizing and producing sequences in the constraint satisfying network. The dynamic system also suggests that a neural network could be further developed to handle more complex information by using further levels of sequence categorizers. The system could also be modified in other ways to enhance some of its capabilities.

In general, the system can be considered to have two main types of components. One is the categorizing parts of the modules that recognize different types of information, either static or sequential, and the other is the associative parts that makes the information from the categorisers useful in some other part of the network. We can thus recognize two types of processes: those that generate information and those that make use of that information. The constraint satisfying network described in this paper is an ideal mechanism for building the correct associations from the categorisers to other parts of the network. These mechanisms can be used in a number of different applications that involve static and sequential information. Processing of linguistic information has been used as an example, but it is likely that the mechanisms involved in many other cognitive processes are of a similar kind.

Some directions for the development of a high level description of neural systems that can be used as the basis for a schema system have also been introduced. This level is more suitable for the description of language learning and processing since it is more general and does not necessarily involve a specific neural model. These ideas has been further developed in Balkenius (1991, 1992).

Time	Input Prediction	Result Read-Out	Sequence Prediction	Syntactic Production
t_1	Mary•Kiss•Pat	Mary•Kiss•Pat	—	—
t_2	John•Kiss•Ag	John•Kiss•Ag	—	—
t_3	—	**John**•Kiss•Ag	(P•Ag, V, P•Pat)	P•Ag
t_4	—	John•**Kiss**•Ag	(V, P•Pat)	V
t_5	—	**Mary**•Kiss•Pat	—	P•Pat

Table 2. Read-out of a sentence after the environmental inputs have been presented. At time t_0 and t_1, two observations of the environment are presented. In this case the two schemata for 'Mary is beeing kissed' and 'John kisses'. At time t_2 a production signal is received by the dynamic system. This causes a read-out of the predicted sequence at $S^{(2)}$ to occur. The read-out excites first the person that is agent then the verb and at last the patient is recalled from *IM* to be read out into CS. The syntactic production at R-O is the result of the prediction from $S^{(2)}$.

REFERENCES

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. 1985. A learning algorithm for Bolzmann machines. Cognitive Science 9, pp. 147-169.

Arbib, Conklin and Hill. 1987. From schema theory to language. New York: Oxford University Press.

Arbib, M. A. and Hanson, A. R. 1987. Vision, brain, and cooperative computation. MIT Press.

Balkenius, C. 1993. Some Properties of Neural Representations. in Bodén, M. Niklasson, L. (eds.) Selected readings of the Swedish conference on connectionism 1992, Ellis Horwood, in press.

Balkenius, C. & Gärdenfors, P. 1991. Non-monotonic inferences in neural networks, in Allen, J., Fikes, R. & Sandewall, E. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference. Morgan Kaufman.

Carpenter, R. H. S. 1984. Neurophysiology. London: Edward Arnold Ltd.

Charniak, E. and McDermott, D. 1985. Introduction to artificial intelligence. Addison-Wesley.

Cohen, M. A. and Grossberg, S. 1983. Absolute stability of global pattern formation and parallel memory storage by competitive nural networks. IEEE Transactions on systems, man, and cybernetics, SMC-13, pp. 815-826.

Cohen, M. and Grossberg. S. 1986. Neural dynamics of speech and language coding: developmental programs, perceptual grouping, and competition for short term memory. Human Neurobiology, 5, pp. 1-22. Springer Verlag.

Cohen, M. and Grossberg, S. 1987. Masking fields: a massively parallel neural architecture for learning, recognition, and predicting multiple groupings of patterned data. Applied Optics, 26 (10), pp. 1866-1891.

Crick, F. 1986. Function of the thalamic reticular complex: the searchlight hypothesis. Proceedings of the National Academy of Sciences 81, pp. 4586-4590.

Edelman, G. 1989. Neural Darwinism. Oxford University Press.

Elman, J. E. 1989. Representation and structure in connectionist models. CRL Technical Report 8903, University of California, San Diego, La Jolla.

Fodor, J. and Pylyshyn, Z. 1988. Connectionism and cognitive architecture:A Critical Analysis, In. S. Pinker & J. Mehler (eds.) Connections and symbols. Cambridge, MA: MIT Press.

Grossberg, S. 1973. Contour enhancement, short term memory, and constancies in reverbating neural networks. Studies in applied Mathematics, LII, pp. 49-57.

Grossberg, S. 1978. A theory of human memory: self-organization and performance of sensory-motor codes, maps, and plans. Progress in Theoretical Biology, Vol. 5, pp. 233-374. Academic Press.

Grossberg, S. 1986. The adaptive self-organization of serial order in behavior: speech, language, and motor control, in Schwab E. C. and Nusbaum H. C. (eds.) Pattern recognition by humans and machines, Vol. 1: Speech Perception. Academic Press.

Hebb, D. O. 1949. The organisation of behaviour. New York: John Wiley and Sons.

Hill, J. C. 1983. A computational model of language acquisition in the two-year-old. Cognition and brain theory, 6(3), pp. 287-317.

Hintikka, J. 1968, in v. Rootselaar, B. and Staal, J. F. (eds.) Logic, Methodology, and Philosophy of Science III, Proceedings of the 1967 International Congress, pp. 311-331. Amsterdam: North-Holland .

Hinton, G. E., and Sejnowski, T. J. 1988. Learning and relearning in Bolzmann machines, in Rumelhart, D. E. (ed.) Parallel distributed processing, Vol 1, pp. 282-317. MIT Press.

Hopfield, J. J. 1982. Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences 79.

Hopfield, J. J. 1984. Neurons with graded response have collective computational properties like those of two-state neurons, Proceedings of the National Academy of Sciences 81.

Kohonen, T. 1988. Self-organization and associative memory. Berlin: Springer Verlag.

Lansner, A. 1986. Investigations into the pattern processing capabilities of associative nets. Dissertation, February, 1986, Department of Numerical Analysis and Computing Science.

von der Malsburg, C. and Bienenstock E. 1986. Statistical coding and short term synaptic plasticity: a scheme for knowledge representation in the brain. in Bienenstock, E. Fogelman-Soulié F. and Weisbuch G. (eds.) Disordered systems and biological organization. Berlin: Springer.

Minsky, M. 1987. The society of mind. London: Heinemann.

Piaget, J. 1952. The origins of intelligence in children. New York. International University Press.

Piaget, J. and Inhelder, B. 1973. Memory and intelligence. London: Routledge & Kegan Paul.

Rumelhart, D. E. and McClelland, J. L. 1986. Parallel distributed processing, Vols 1 and 2, MIT Press.

Rumelhart, D. E. and Kawamoto, A. H. 1986a. Mechanisms of sentence processing: assigning roles to constituent of sentences. in Rumelhart, D. E. and McClelland, J. L. (eds.) Parallel distributed processing, Vol 2, pp. 272-325. MIT Press.

Rumelhart, D. E. and McClelland, J. L. 1986b. On learning past tenses of english verbs. in PDP models. in Rumelhart, D. E. and McClelland, J. L. (eds.) Parallel distributed processing, Vol 2, pp. 216-271. MIT Press.

Rumelhart, Smolensky, McClelland and Hinton. 1986c. Schemata and sequential thought processes. in PDP models. in Rumelhart, D. E. and McClelland, J. L. (eds.) Parallel distributed processing, Vol 2, pp. 7-57. MIT Press.

Rumelhart, D. E., and Zipser, D. 1988. Feature discovery by competitive learning. in Rumelhart, D. E. and McClelland, J. L. (eds.) Parallel distributed processing, Vol 1, pp. 151-193. MIT Press.

Schank, R. and Abelson, R. P. 1977. Scripts, plans, goals, and understanding. Lawrence Erlbaum, N.J: Hillsdale.

Smolensky, P. 1986, Information processing in dynamical systems: foundations of harmony theory. in Rumelhart, D. E. and McClelland, J. L. (eds.) Parallel distributed processing, Vol 1, pp. 194-281. MIT Press.

Woods, W. A., 1975, What's in a link: Foundations for semantic networks. in Bobrow, D. and Collins, A. (eds.) Representation and understanding. New York: Academic Press.