Rule-Based Processing in a Connectionist System for Natural Language Understanding

Bart Selman

Technical Report CSRI-168
January 1985

Computer Systems Research Institute
University of Toronto
Toronto, Canada
M5S 1A1

The Computer Systems Research Institute (CSRI) is an interdisciplinary group formed to conduct research and development relevant to computer systems and their application. It is jointly administered by the Department of Electrical Engineering and the Department of Computer Science of the University of Toronto, and is supported in part by the Natural Sciences and Engineering Research Council of Canada.

Rule-Based Processing in a Connectionist System for Natural Language Understanding

Bart Selman

Technical Report CSRI-168

April 1985

The Computer Systems Research Institute (CSRI) is an interdisciplinary group formed to conduct research and development relevant to computer systems and their application. It is jointly administered by the Department of Electrical Engineering and the Department of Computer Science of the University of Toronto, and is supported in part by the Natural Sciences and Engineering Council of Canada.

Rule-Based Processing in a Connectionist System for Natural Language Understanding

Bart Selman
January 1985

A Thesis submitted in partial fulfillment of
the requirements for the Degree of Master of Science

Department of Computer Science
University of Toronto
Toronto, Canada

Abstract

In this thesis we give a connectionist model for natural language processing. In contrast with previously proposed schemes, this scheme handles traditionally sequential rule-based processing in a general manner in the network. Another difference is the use of a computational scheme similar to the one used in the Boltzmann machine (Fahlman et al. 1983). This allows us to formulate general rules for the setting of weights and thresholds.

We give a detailed description of a parsing system based on context-free grammar rules. Using simulated annealing, we show that at low temperatures the time average of the visited states at thermal equilibrium represents the correct parse of the input sentence.

The system is built from a small set of *connectionist primitives* that represent the grammar rules. These primitives are linked together using pairs of computing units that behave like discrete switches. These units are used as binders between concepts. They can be linked in such a way that individual rules can be selected from a collection of rules, and are very useful in the construction of connectionist schemes for any form of rule-based processing.

We also consider two variations on the formalism of the Boltzmann machine. First we show how the use of $+1$ and -1 as output values for the computing units facilitates the setting of thresholds. Secondly we introduce an alternative energy function. Using this function we are able to choose a set of weights, such that only the state of the network that corresponds to the correct parse of the input sentence has the lowest possible energy.

Acknowledgements

Firstly I would like to thank my supervisor Graeme Hirst for giving me continuous support and guidance throughout my thesis work. I am also very grateful to my second reader, John Mylopoulos, for reading the thesis and giving useful criticism.

I thank Dana Ballard, Garrison Cottrell, Jerry Feldman, Geoffrey Hinton, and Mike Luby for useful discussions and comments, and Jeff Air, Maya Guha, Graeme Hirst for proofreading the thesis.

Financial assistance was gratefully received from a Government of Canada Award and from the Department of Computer Science.

Table of contents

CHAPTER 1

Introduction

Recently, work on parallel architectures has become a major part of research in artificial intelligence (Fahlman, Hinton and, Sejnowski 1983; Feldman and Ballard 1982). Such architectures consist of a large number of processing elements, also called computing units, which work on a single task. An important motivation behind this research is the belief that the application of parallel architectures will lead to the development of algorithms that differ fundamentally from existing sequential algorithms and are better suited to handle cognitive tasks. In chapter 2 we will give an overview of existing computing models for parallel architectures for artificial intelligence (AI) and discuss their performance with respect to some typical AI tasks.

Most of the research on these architectures concentrates on low-level vision tasks. Some research, though, has been done on the application of parallel architectures in other AI domains, like natural language understanding. Pollack and Waltz (1984) and Small and Cottrell (1983) give parallel processing schemes, based on the deterministic continuous connectionist scheme (McClelland and Rumelhart 1981; Feldman and Ballard 1982), which handle word-sense and syntactic disambiguation. Reilly (1984) uses a similar scheme for anaphora resolution. A central aspect of these models is that they process the different sources of knowledge used in NLU, such as lexical and world knowledge, in a highly integrated way; for example the syntactic and semantic processing are integrated. The work on these models has been very promising with respect to the disambiguation task. This might be explained by the fact that connectionist models are closely related to discrete marker passing systems, which have proven to be useful in disambiguation tasks (Hirst 1983).

In chapter 3 we will discuss the major existing schemes. Instead of concentrating on their strong point, disambiguation, we will focus on the major limitation of these schemes; a very limited capability for tasks such as parsing and case filling which seem to require processing to be based on a set of rules. For example, Small and Cottrell (1983) give a network for parsing the sentences "Bob threw up a ball" and "Bob threw up dinner". In this network 'Bob' is only linked to the agent case and not to the object case, so the fact that 'Bob' is likely to be the agent of the sentence is directly implemented in the network. We propose a more general network, where

'Bob' is linked to both the agent and the object case using two intermediate units, called binder units. In this scheme syntactic information is used to activate one of the binder units to indicate the case role of 'Bob' in the particular input sentence.

In chapter 4 we will present a detailed connectionist system for rule-based processing. This system parses an arbitrary input sentence up to a given length using context-free grammar rules. Schemes for rule-based processing, integrated with schemes that concentrate on disambiguation might lead to a general connectionist model for NLU. Such a model should also incorporate ways to directly store knowledge, and be able to use this knowledge for the processing of other sentences. We will not address this problem in detail.

A problem that we will discuss in detail is the setting of weights and thresholds in a connectionist scheme (chapter 4). In work done so far on connectionist models for NLU, this problem has not been given much consideration. However, it became clear to us from running simulations of some connectionist models that the choice of the set of weights and thresholds is far from trivial and that the performance of the model is highly dependent on the chosen weights. In chapter 4 we will give general rules for the setting of weights and thresholds in our parsing scheme. These rules are based on symmetry considerations and the analysis of parts of the network in isolation. We will consider an example network, with weights and thresholds set according to these rules, and give simulation results showing that the network finds the correct parse for a set of sentences used as test data. We will also describe the effect of certain changes in weights and thresholds on the performances of this network.

In our system we use the computational scheme of the Boltzmann machine (Hinton and Sejnowski 1983). This stochastic scheme has some formal properties which make it easier to analyze than the deterministic continuous scheme (Feldman and Ballard 1982). In chapter 4 we will discuss the advantages of this stochastic scheme over the deterministic one.

In chapter 5 we will consider two modifications of the computational scheme of the Boltzmann machine. One modification is of practical interest; it facilitates the implementation of symmetrical interdependency relations between hypotheses in the connectionist scheme. The other modification is of theoretical interest; it allows us to prove that for a certain set of weights and thresholds the network will always give a consistent global behavior, that is, it will find the correct parse of the input sentence (provided of course that the sentence is part of the language defined by the grammar).

Our motivation behind this research is twofold. On one hand we believe that, at least part of the natural language understanding process can be handled by a connectionist architecture and that this form of integrated, parallel processing facilitates the parsing process. On the other hand, we believe that these schemes can only be of practical interest for NLU if they are able to handle rule-based processing, like syntactic parsing, in a general and efficient way and are understood well enough to set the weights and thresholds correctly in large networks.

CHAPTER 2

Massively parallel architectures for AI

2.1 Introduction

In this chapter we will discuss the current research on parallel computer architectures for artificial intelligence. As an introduction, we will look at the motivation behind the current interest in parallel architectures from three different perspectives: anatomy, computational complexity, and technology.

Von Neumann (1958) points out that large and efficient natural automata (like the human brain) are highly parallel, while large and efficient artificial automata (like the Von Neumann machine) are more serial. He shows that the choice of a particular architecture has far-reaching consequences upon the form of optimal algorithms, computing time, and required storage space.

Currently there are two different approaches to research on new parallel architectures. One approach is that of the connectionist models in cognitive science (McClelland and Rumelhart 1981; Feldman and Ballard 1982; Fahlman, Hinton, and Sejnowski 1983). These models make an attempt to model the actual behavior of the neural-net that constitutes the animal brain. Another approach, which resulted for example in marker-passing systems (Fahlman 1979), applies parallelism to carry out cognitive tasks but does not explicitly attempt to model the neural-net of the brain.

Let us now consider the aspect of computational complexity. The computations in the animal brain are carried out by relatively slow (milliseconds) neural computing elements in a complex network. Relatively complex cognitive tasks are carried out in a few hundred milliseconds (Posner 1978). This means that such tasks are carried out in less than a hundred time steps. Current AI programs for similar tasks require millions of time steps on conventional serial computer architectures. Apparently time is the critical factor in the use of those programs. Therefore, although basic automata theory shows that the difference between parallel and serial processing is not a fundamental one, computation time can have a tremendous effect on the style of

computing that is chosen.

There are many operations and algorithms that are computationally quite feasible on a parallel machine but would never be seriously considered on a serial machine because they would take too long. An example of such an operation is the intersection of two large sets, which can be very useful in recognition tasks. On a serial machine we go to great lengths to avoid this operation, as it takes time proportional to the size of the sets. On some parallel machines, as we see later, it takes only a few cycles, regardless of the size of the sets.

One of the factors that will determine whether the work on parallel architectures will have a direct impact on AI research is the availability of the technology to build such architectures. It seems that the rapid developments in VLSI techniques will make it feasible to build those architectures in the near future. Up to now none of the proposed vast parallel machines have actually been built. Their properties have been studied using simulations on serial machines.

One useful way to classify parallel architectures is by the type of signals that are passed among the elements. Fahlman (1982) proposes a division of these systems into three classes: message-passing, marker-passing, and value-passing architectures.

Message-passing systems form the most powerful family. They pass messages of arbitrary complexity, and perform complex operations on these messages. In such networks there may be severe contention and traffic congestion problems. Messages passed between neurons are of limited complexity (Feldman and Ballard 1982); therefore message-passing systems are not suited to modeling information processing in the brain. Message-passing systems are in general referred to as distributed computing systems. Applications of these systems in AI are a topic of current research; see for example Lesser and Corkill (1983). Since, up to now, no applications in natural language understanding have been studied, we won't discuss message-passing systems in this thesis.

Marker-passing systems are the simplest class of parallel architecture, and the most limited. Communication among the computing elements is in the form of markers. A marker consists of a small group of bits. Each unit can store a limited number of markers and can combine them using simple Boolean operations. In section 2.2 these systems are discussed in more detail.

Value-passing systems pass continuous quantities or numbers and perform simple arithmetic operations on them. An interesting subclass of these systems are the connectionist systems. Connectionism refers to the fact that all the knowledge in these systems is stored in the connections (links) between the computing units. Con-

nectionist models are sometimes referred to as relaxation models, because the way they compute is similar to that of solving partial differential equations using the relaxation method. These models will be discussed in section 2.3.

2.2 Marker-Passing Architectures

The best-known example of a marker-passing system is NETL, developed by Fahlman (1979, 1981). It was designed to be implemented in hardware, although this goal has not yet been achieved. NETL is representative of the family of marker-passing systems. In this section we will give a short description of how NETL operates.

The human mind can store a tremendous quantity of knowledge and can access whatever knowledge it needs very quickly and in a flexible manner, as will be illustrated with an example later on. An AI system, if it is to serve as a model for human knowledge-handling abilities, must exhibit comparable capacity, speed, and flexibility. Because much of the information we use in everyday life is not stored explicitly but must be deduced from other information, a seemingly simple query may give rise to a very substantial amount of deduction and search.

Suppose, for example, that we tell you that Clyde is an elephant. Informed of this simple fact, you will know considerably more about Clyde than explicitly told. You will be able to tell us, for example, how many legs Clyde has, what color Clyde is, whether Clyde can fly etc. (all with a fair degree of certainty). It appears unlikely that most of this knowledge is stored explicitly. A more likely explanation is that most of the above questions are answered by using some form of deduction from the fact that Clyde is an elephant. Therefore, in an AI system we store the name Clyde in the knowledge base with a link to the concept elephant. In this case the link represents an IS-A relation (see Winston 1984). Certain properties of Clyde, for example his color, follow directly from the general properties of an elephant. To find these properties, we have to search the knowledge base. Figure 2.1[1] gives a portion of a possible hierarchy of which Clyde is a member.

An exhaustive search in a large AI system based on a serial model requires, in most cases, an unacceptable amount of time, especially if the IS-A hierarchy is allowed to branch upwards and downwards and is very broad. The search algorithm has to visit sequentially a large number of nodes at each level, and this number grows rapidly as one gets farther away from the starting level. Therefore one must provide certain heuristic rules to guide the search process. A problem with these rules is that they are strongly-domain dependent, so one must provide specific rules for each

[1] The figures 2.1, 2.2, and 2.3 were taken from Fahlman (1981).

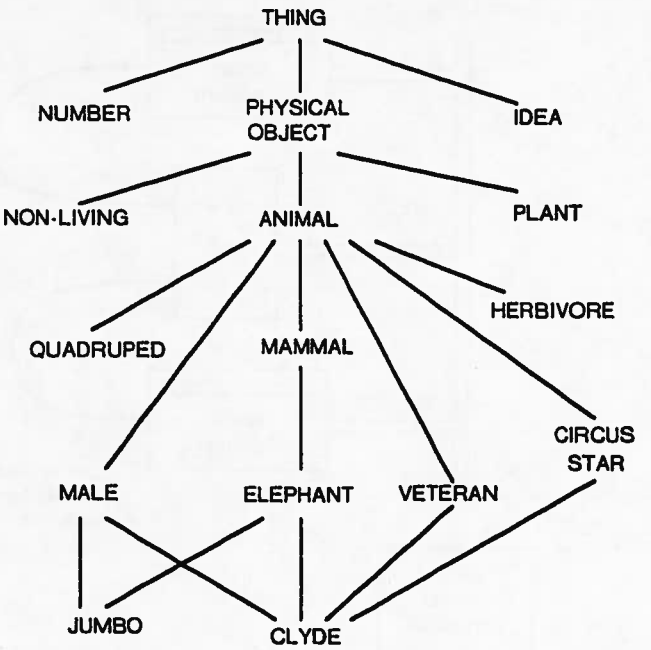


Figure 2.1 A portion of an IS-A hierarchy containing Clyde.

knowledge domain. NETL provides an exhaustive search mechanism with a search time proportional to the depth of the hierarchy in which the knowledge is stored.

The basic components of the NETL system are given in figure 2.2. In NETL, concepts are represented by very simple hardware computing units. Relationships between the concepts and simple assertions about those relationships are represented by additional hardware elements called links. There are different types of links, each type representing a certain relation, for example IS-A, PART-OF or COLOR-OF. (The A- and B-wire are just labels for convenient reference to the figure.) The computing units and the links are all attached through a shared, party-line bus to an external controller, a conventional serial computer. The controller is able to "shout" simple commands to all of the units and links together, or to specified subsets of them, or to individual units and links for which it knows the name or the serial number. Selected units are able to reply by placing their own name or serial number on the shared bus. The units can send 16 different markers (each marker consists of four bits) over the wires.

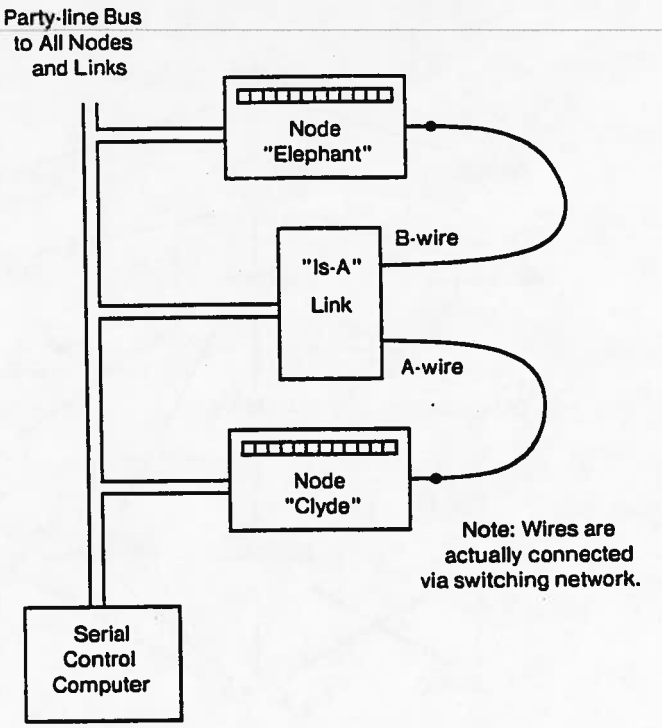


Figure 2.2 The basic hardware components of NETL.

A simple example illustrates how this machinery can handle exhaustive search and deduction. Consider the hierarchy given in figure 2.3, and suppose we want to determine the color of Clyde. First, the control computer tells the Clyde node to set marker #1 on its surrounding wires. Next, all IS-A links that sense marker #1 on their A-wire are told to place a copy of marker #1 on their B-wire. In these two operations we have simultaneously marked all units one level above Clyde in the IS-A hierarchy. (Note that the use of the A-wire and B-wire prevents the markers from going downwards.) These two operations are repeated until the top unit, called THING, receives a marker #1. At this point we have marked all units above Clyde in the IS-A hierarchy, that is all the units from which Clyde is supposed to inherit properties. We now send a command to all COLOR-OF links with a marker #1 on their A-wire, telling them to send marker #2 to the unit tied to the B-wire. Finally, we tell all units with marker #2 to place their names on the bus. There will be one such unit, GRAY, the desired answer. This example illustrates one of the strongest points of a marker-passing system, namely a very efficient computation of the closure of transitive relations. In section 2.4 we will consider several other computational problems, and discuss whether they can be handled by a marker-passing system.

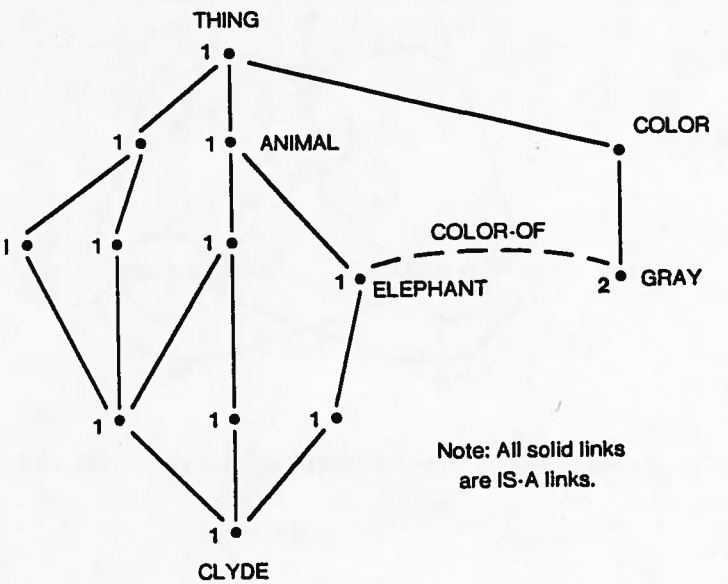


Figure 2.3 Finding the color of Clyde.

NETL has never been implemented in hardware. The problem is not to implement computing units and links, because with current VLSI technology it is feasible to put a thousand or so of these elements on a single chip. The real difficulty lies in having to wire together new nodes and links as new knowledge is acquired. Therefore, a large switching network between the units and the nodes is necessary. The design of this network is very complicated. Up to now, NETL has been studied using simulations on serial computers. On a PDP-10, for example, such simulations are limited to about 10,000 elements, not enough for real-world applications.

2.3 Connectionist Models

2.3.1 Introduction

We will first define the variables that will be used in the description of the various types of connectionist models. Figure 2.4 gives the general form of a connectionist model consisting of three computing units. Each unit has several inputs and one output. The units are numbered, to be able to refer to specific ones. They are linked together, and each link has a certain weight (strength of connection). One of the inputs of a unit is not linked to any output. This external input, η, is used to provide the system with information from outside. Each unit also has a certain threshold θ. Input values below this threshold value don't have any influence on the system.

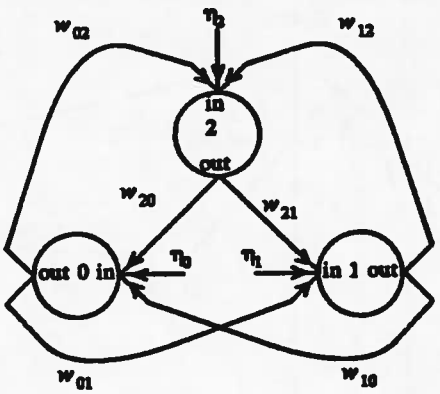


Figure 2.4 The general form of a connectionist system consisting of three computing units.

The output of a unit at time $t+\Delta t$ is a function of the output at time t, the set of weighted input values, the external input, and the threshold of that unit at time t. Thus the output value of the j^{th} unit is given by

$$s_j(t + \Delta t) = f\ (s_j(t), \{w_{ij}s_i(t)\}, \eta_j(t), \theta_j), \qquad (2.1)$$

where s_j is the output of the j^{th} unit with threshold θ_j, w_{ij} is the weight of the link between the output of the i^{th} unit and the input of the j^{th} unit, η_j is the value provided to the external input of the j^{th} unit. We will only consider relaxation schemes with discrete time steps; therefore we assume that Δt is a fixed finite value. The state of the system at time t is defined as the set of output values at time t, $\{s_i(t)\}$.

A connectionist system computes in the following way. Assume the system is in a state $\{s_i(0)\}$ at $t = 0$ and we provide the system with input data, $\{\eta_i(0)\}$. Consequently the system computes the next state $\{s_i(\Delta t)\}$ using (2.1). If we now provide the system with the next set of input values, the system can compute the state at time $2\Delta t$, etc. When the function f is deterministic, and we provide the system with a static set of input values it will eventually settle into a *stable state*[1] (a state in which the system will stay forever, provided that the input is constant). Which stable state the system reaches depends on the input data, the initial state, and the set of weights $\{w_{ij}\}$. When the function f is non-deterministic, the system will reach an 'equilibrium'; that is, it will visit a set of states such that the frequency with which it visits a particular state in this set is time-independent.

---

[1] We assume that the system does not oscillated.

It is usual to consider only systems with symmetrical connections between the units, so that for all links

$$w_{ij} = w_{ji} \; . \qquad (2.2)$$

Figure 2.5 gives a simple, and often-used representation of a symmetrical connectionist system consisting of three computing units. In this diagram the actual inputs and the output of the computing units are not shown.

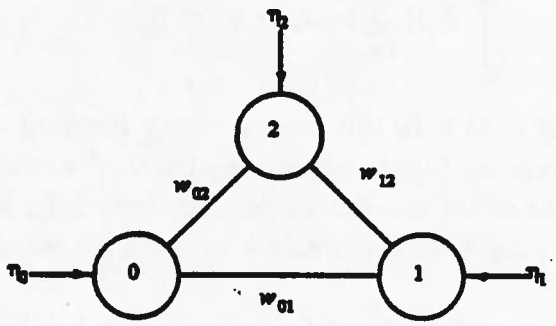


Figure 2.5 The system given in figure 2.4 with symmetrical connections (for example $w_{02} = w_{20}$).

We have now defined the general framework of the connectionist model; the specific choice of the function f in equation (2.1) will define the type of connectionist model. Hopfield (1982) uses a deterministic, binary function. We will refer to his model as the *determinitic binary model*. Fahlman, Hinton and Sejnowski (1983) use a stochastic, binary function in their *Boltzmann machine*. And, McClelland and Rumelhart (1981; Feldmann and Ballard 1982) use a deterministic, continuous function. We named this model the *deterministic continuous model*. These models will be discussed below.

Before doing so we would like to point out, that although there are superficial similarities between connectionist models and perceptrons (Minsky and Papert 1968), the connectionist models are much more powerful (in a computational sense) because they allow feedback, while in perceptrons activation flows only in one direction. An example of the limited computing power of perceptrons is the fact that they can't determine whether or not all parts of a geometric figure are connected to one another.

2.3.2 The deterministic binary model

The dynamic behavior of a deterministic binary system (Hopfield 1982) is given by[1]

$$s_j(t+\Delta t) = \begin{cases} 1 \text{ if } \sum_{i \neq j} w_{ij} s_i - \theta_j > 0 \\ 0 \text{ if } \sum_{i \neq j} w_{ij} s_i - \theta_j \leq 0 \end{cases} \qquad (2.3)$$

So the function f in (2.1) is in this case a binary function, and the computing units have binary output values. Input data is supplied to the model by keeping the output of a set of units (*input units*) at a fixed value; so these units won't take part in the updating process ('clamping' the output values of the input units).

We will call the computing unit j *active* if $s_j = 1$; otherwise the unit is *inactive*. Other terminology, introduced by Hinton and Sejnowski (1983a), will facilitate the discussion of this model with regard to AI applications. They refer to a computing unit as an *hypothesis*: if $s_j = 1$ the hypothesis associated with the j^{th} unit is true; otherwise it is false. The weights on the links between the units are referred to as *constraints* between the hypotheses.

First we will consider a simple example which shows how knowledge can be stored in the weights. Assume we have a deterministic binary connectionist system in which unit #1 stands for the hypothesis 'John is a dog' and unit #2 stands for the hypothesis 'John is a cat', and the name 'John' refers to one object. We can store the knowledge 'John can't be both a dog and a cat' by assigning large negative values to the weights w_{12} and w_{21}. From (2.3) we see that, given these weights, it becomes very unlikely that both s_1 and s_2 are equal to 1.

Another example will show us how this parallel system can be used to perform a cognitive task, namely that of recognizing an object from partial input data. Consider a machine consisting of nine units and a knowledge base containing two objects A and B. Instead of associating each object with just one computing unit, as we did in section 2.2, we associate the presence of each object with a pattern of active and inactive units. These patterns are given in figure 2.6a. This form of representation is called the *distributed* or *global* representation (versus the *local* representation; one

---

[1] In this model the computing units update asynchronously; Δt is the average time between updates.

unit for each object, as in the previous example). Figure 2.6b shows a possible set of weights which represents the objects A and B in the system (each unit has a small negative threshold to prevent spontaneous activation). Now assume the machine is in an arbitrary state and we supply the partial information: $s_0 = 1$ and $s_8 = 1$. After a number of time steps, it follows from equation (2.3) that the output of the unit s_4 will become 1 and the other outputs will become 0. Thus the system has found the best match between the input data and the knowledge in the system, namely object B.

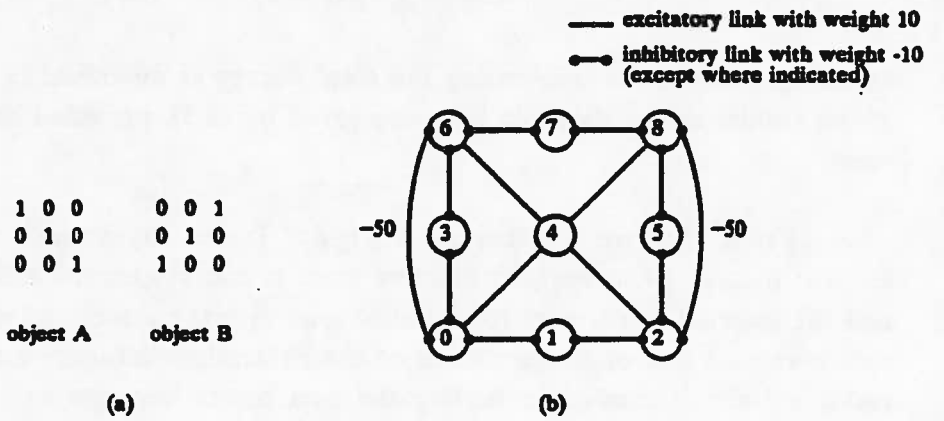


Figure 2.6 Part (a) shows the distributed representations of objects A and B. Part (b) gives an example of a connectionist network in which these concepts are stored. (Only the connections with non-zero weights are drawn.)

The given examples only illustrate the ideas behind the model; the actual power of the model appears only when one considers large systems. A problem with larger systems is to determine the set of weights representing the knowledge in the system. To determine these values in an efficient way one must analyze the dynamic behavior of the system. The dynamic behavior, as given by (2.3), is very hard to analyze for time-varying input data. But in the case of constant input data it can be shown (Hopfield 1982) that the system behaves in such a way that it minimizes a global quantity E given by

$$E = -1/2 \sum_{ij} w_{ij} s_i s_j + \sum_i \theta_i s_i, \qquad (2.4)$$

Because this expression is very similar to the expression for the energy of binary models in solid state physics (see for example the Ising model, Binder 1978), the quantity E is called the energy of the system.

A simple way to find a local minimum of the energy is to repeatedly switch each unit into whichever of its two states yields the lower total energy given the current state of the other units. When the system reaches a local minimum, no unit will be switched any more, because such a switch would increase the total energy. Apparently a local minimum of the energy can be associated with a stable state (provided the constant input data). From (2.2) and (2.4) it follows that the energy difference ΔE_j between a state with $s_j = 0$ and a state with $s_j = 1$ is given by

$$\Delta E_j = \sum_i w_{ji} s_i - \theta_j \qquad (2.5)$$

So the procedure for minimizing the total energy as described in the preceding paragraph results in the dynamic behavior given by (2.3), provided the input data is constant.

From (2.4) we see that the energy E is actually a measure of how badly the current pattern of active and inactive units in the system fits the external input data and the internal constraints (the connections with the associated weights). This observation reveals one of the problems of the deterministic binary model. In recognition tasks, we are interested in finding the best match between external input data and internal constraints, that is, we want to find the global minimum of the energy. However, when the system first reaches a local minimum it will never find the global minimum (because the deterministic system can only decrease its energy, see (2.3) and (2.5)). In the next section we will discuss a model in which this problem does not occur because it can escape from a local minimum and continue searching for a global minimum. This is achieved by introducing a stochastic component in the system.

2.3.3 The Boltzmann Machine

In the Boltzmann machine, Hinton and Sejnowski (1983a, 1983b) introduce a non-deterministic component in the updating scheme of the deterministic binary model, to allow the system to escape from a local minimum. They introduce a probability p_j given by

$$p_j = \frac{1}{1 + e^{-\Delta E_j / T}} \qquad (2.6)$$

in which ΔE_j is given by equation (2.5) and T is a formal parameter denoting the computational temperature, whose role will be explained below. Using the probability p_j the dynamic behavior of the Boltzmann machine is given by

$$s_j(t+\Delta t) = \begin{cases} 1 \text{ with a probability of } p_j \\ \\ 0 \text{ with a probability of } 1-p_j. \end{cases} \qquad (2.7)$$

This decision rule is the same as the one that determines the thermodynamic behavior of a physical system consisting of particles which have only two energy states. At thermal equilibrium the probability distribution of the global states of such systems is given by the Boltzmann distribution. So, at thermal equilibrium for each pair of states α and β of a network using updating rule (2.7) the following equation holds:

$$\frac{P_\alpha}{P_\beta} = e^{-(E_\alpha - E_\beta)/T}, \qquad (2.8)$$

in which P_α is the probability that the system will be in state α with energy E_α. Just like 'real' physical systems, the networks will at each temperature eventually reach a thermal equilibrium.

From (2.6) and (2.7) it follows that when T approaches zero the Boltzmann machine becomes a deterministic binary connectionist machine; equation (2.7) reduces to (2.3). On the other hand, if T approaches infinity, the state of the machine will be a random pattern of active and inactive units, because each unit has a chance of 0.5 to become active. So apparently at very low temperatures the system will not be able to increase its energy and therefore the system can't escape from local minima; while at very high temperatures the state of the system will become independent of the energy and therefore independent from internal constraints in the system.

The computational scheme used in the Boltzmann machine is a special form of the *Monte Carlo* method (Binder 1978), and is often used in statistical mechanics to study multi-variable energy functions. Kirkpatrick et al. (1983) describe how one can find the minimum value of an multi-variable function using this form of the Monte Carlo method, namely by starting at a high temperature and subsequently lowering the temperature to zero (when T approaches the freezing temperature of the system the temperature should be lowered very slowly). This method is called *simulated annealing* because of its analogy with the growing of 'perfect' crystals. Here, one starts in the liquid state and lowers the temperature. When one approaches the freezing point the temperature should be lowered very slowly until finally all the material has crystallized. The perfect crystal state is the state with minimal internal energy.

Hinton and Sejnowski (1983a) apply a simulated annealing scheme in the Boltzmann machine, but do not lower the temperature below $T = 1$. In this way each hypothesis (or computing unit) can still change its state; the average time spent in the true state is a measure of the belief in the truth of the hypothesis. The following example illustrates the advantage of this approach.

Consider the Necker cube in figure 2.7. In a connectionist system there will be two alternative collections of hypotheses that form equally plausible interpretations of this figure. The Boltzmann machine at $T = 1$ will occasionally flip between these two alternatives (spending on the average half of the time in one of the two alternative states). However, when we further lower the temperature to zero the system will settle down in one of the two alternatives, and the other plausible interpretation would be hidden in the system.

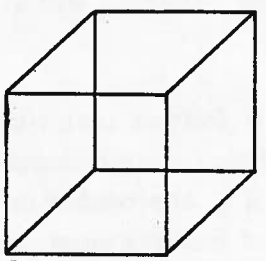


Figure 2.7 The Necker cube, showing two equally plausible interpretations.

Hinton and Sejnowski (1983a, 1983b, 1984; Hinton, Sejnowski, and Ackley 1984) give a learning procedure for the Boltzmann machine. The procedure requires a set of pairs of inputs and desired outputs (a *training set*) to train the system. During the training period, the system finds the regularities in the training-set and stores this knowledge by adjusting its weights. In general, it is difficult to find an appropriate training set, but in low-level vision applications this task is relatively easy. So far, no learning algorithms have been found for the other types of connectionist models.

Another positive feature of this model is that there is an equation that relates the weight on a link between two units, each representing an hypothesis, and the inter-dependency of these hypotheses expressed in terms of conditional probabilities. We will discuss this in more detaili in section 5.3.

2.3.4 The deterministic continuous model

In the deterministic continuous model (McCLelland and Rumelhart 1981; Feldman and Ballard 1982), the output of a computing unit is a continuous value on a fixed interval; a common choice is an output value between −10 and +10. The dynamic behavior of a commonly used version of this model is given by

$$s_j(t+\Delta t) = s_j(t) + \sum_{i \neq j} w_{ij} \lfloor s'_i(t) \rfloor \qquad (2.9)$$

in which,

$$s'_i(t) = \begin{cases} s_i(t) - \theta_i & \text{iff } s_i(t) > \theta_i \\ 0 & \text{iff } s_i(t) \leq \theta_i \end{cases} \qquad (2.10)$$

In the application of this model, the value of the output of a unit is used as a measure of the belief in the truth of the hypothesis which is represented by the unit.

The fundamental premise of all connectionist models is that computing units do not exchange large amounts of information (this is based on the observations that individual neurons seem to transmit only small amounts of information). The entier function in (2.9) is needed to assure that the deterministic continuous model satisfies this fundamental premise; in each time step, only $\log_2 21$ bits of information need to be transferred between two units (there are 21 integers on the interval [-10, +10]).

An important difference between this model and both the deterministic binary model and the Boltzmann machine is that in this case there is no known global quantity which is being minimized by the system. Therefore the behavior of the system is very hard to analyze (even in the case of a constant input). This makes it difficult to choose an appropriate set of weights to store knowledge in the system. So far, no efficient procedure (for example a learning method) to determine this set has been developed.

2.4 Computational problems

In this section we will discuss briefly the use of parallel architectures in some computational problems. We will only consider operations that are relevant for AI applications. A more detailed discussion can be found in Fahlman et al. (1983).

Set-intersection. Pattern recognition in its simplest form can be viewed as a set-intersection problem. Therefore, we can associate with each observable feature a set of objects that exhibit this feature, which we will call the object set. When given a number of features, we want to find the item(s) that exhibit all those features; that is, we have to intersect all the object sets associated with the given features. On a serial machine this operation takes time proportional to the size of the smallest set. On a marker-passing machine this operation requires only one step, once the members of each set are marked with a different marker. The system simply asks for the units that have collected all the markers.

A connectionist system can also carry out set-intersection in one step, after marking all the members of each set with a small amount of activity. The system asks for all units with activation above a certain threshold. For the deterministic binary machine and the Boltzmann machine this operation is less straightforward. Hinton (1981) describes a method for doing set-intersection with the Boltzmann machine, at least in simple cases.

Transitive closure. In a knowledge-based system, one frequently wants to compute the closure of certain transitive relations (for example the IS-A relation). On a serial machine this operation takes time proportional to the size of the answer set. In a marker-passing system it takes time proportional to the longest chain of relations that has to be followed, as we saw in section 2.2. A deterministic connectionist systems, using the local representation, can simulate the marker-passing systems in this task, which leads to a similar performance. (Marker-passing can be simulated by activating the set of units of which one wants to determine the transitive closure; subsequently one follows the activation which is spreading in the system.) So far, no general method for computing the transitive closure with the Boltzmann machine has been developed.

Best-match recognition. The recognition procedure described above requires that the features be discrete, noise-free, and that every member of a class exhibits all the associated features. In a real-world recognition task, these requirements are seldom fulfilled. Thus, a perfect match between given features and a stored description does not exist in most recognition problems. In such cases one is interested in the best-match. Marker-passing systems handle this problem very poorly, whilst a connectionist machine handles this task very well. In a connectionist network each observed feature (represented by one or more units) sends activation to a number of units. The amount of activation depends on the level of confidence in the observed feature and the weight(s) of the link(s) between the unit(s) representing the feature and the unit(s) representing some object. After a number of time steps a continuous deterministic system will reach a stable state, that represents the best match between the given features and the stored description. In the Boltzmann and the deterministic

binary system, the best-match is given by the state with the lowest energy.

Gestalt recognition. Gestalt recognition is basically a search mechanism which combines top-down and bottom-up processing to find the best-match between given features and stored descriptions (in the discussion on best-match recognition we considered only the bottom-up processing). Palmer (1975) argues that gestalt recognition is useful in real-world recognition problems: the whole object can only be identified on the basis of its features, but the features can only be identified in relation to one another and to the emerging picture of the whole. If taken out of the context, each feature is ambiguous. A very similar problem occurs in word sense disambiguation in NLU; the meaning of a text can only be determined on the basis of the meaning of its words, but the meaning of the ambiguous words can only be determined in relation to one another and to the emerging meaning of the whole text. The Boltzmann machine handles this task in an elegant way. The observations provide the input to the machine. The set of observed features forms the input to the system and the set of weights represents stored descriptions. To determine the optimal set of weights a learning method as given by Hinton and Sejnowski (1983a) can be used when a training set is available. The energy function over the possible states of the system is a measure which combines these sources of information. Thus, in the search for the global minimum in the energy (the best match) both sources of information are simultaneously used. Using a deterministic binary system, there is a risk of reaching a local minimum, from which the system can't escape. A deterministic continuous connectionist system is in principle able to handle gestalt recognition tasks, but it will be difficult in large systems to find the optimal set of weights to store this set of descriptions.

CHAPTER 3

Connectionist systems for natural language understanding

3.1 Introduction

In this chapter we will discuss the use of parallel architectures in natural language understanding (NLU). First, in section 3.2, we will give an overview of previously proposed connectionist models for NLU. These models show how connectionism can be used to handle several ambiguity problems in NLU. They are also intended to be cognitive models of the way people deal with ambiguity.[1]

In our work the emphasis has been on extending the natural language processing capabilities of these models; we are not making explicit claims about the psychological validity of our approach (except in so far as the connectionist model itself is a cognitive model of human information processing). Before we introduce our model in section 3.4 we will first consider the major shortcoming of the previous models, namely their limited capacity of rule-based processing, in section 3.3.

3.2 Disambiguation

3.2.1 Introduction

An important aspect of any natural language processing system is its ability to disambiguate. Consider, for example, lexical ambiguity; Gentner (1982) found that the 20 most frequent nouns have an average of 7.3 senses each, and the 20 most frequent verbs have an average of 12.4 senses each. It is interesting to note, that people are able to handle lexical disambiguation quite easily, even though the task requires knowledge from many sources.

---

[1] It should be noted that we interpreted the term disambiguation in a broad sense; that is we treat for example anaphora resolution as a disambiguation problem.

Most research on marker passing and connectionist models in NLU, so far, has been concerned with lexical disambiguation. The main reason for this is that both marker passing and connectionism provide a good mechanism to find semantic associations between concepts; semantic associations are relevant in lexical disambiguation, because of a phenomenon called *semantic priming*. Psycholinguistic research shows that semantic priming speeds up a person's lexical disambiguation. Semantic priming refers to the phenomenon that the activation of concepts in the brain speeds up the reaction time for judgements involving associated concepts (Collins and Loftus 1975). An example of semantic priming was found in the study of partial reading by McClelland and Rumelhart (1981). They show that the presence of a printed letter in a brief display is easier to determine when the letter is presented in the context of a word than when it is presented by itself.

It should be noted that marker passing is studied for its use as a separate component of conventional NLU systems to facilitate disambiguation. In the connectionist approach our final goal is an independent connectionist system that handles many aspects of natural language processing. The main motivation behind this approach is the belief that natural language processing should handle different sources of knowledge, like syntax, semantics and pragmatics, in an integrated manner.

3.2.2 Marker passing

The idea to use marker passing to find semantic associations between concepts was introduced by Quillian (1968). Quillian's Teachable Comprehender (1969) uses the concept of semantic association to resolve some simple disambiguation problems. His work demonstrated the applicability of marker passing in disambiguation tasks. Since then, this technique has become much more sophisticated; as an example we will consider a recent system developed by Hirst (1983).

Hirst introduces a general mechanism to handle disambiguation: *Polaroid Words*. Polaroid Words are embedded in a NLU system developed at Brown University, and use marker passing as a disambiguation tool. The NLU system uses the Frail knowledge representation language which contains a built-in marker passer (Charniak, Gavin and Hendler 1983). Strictly speaking, the marker passer in Frail is a message-passing system, because each marker consists of a list of names of nodes that have already been marked. However, because only the length of a path is used for the lexical disambiguation, we will call the system a marker passing system.

Frail is a frame-based knowledge representation language. Given the name of a node (a frame, a slot, or an instance) in the knowledge base, as a first step the marker passer marks all nodes directly linked to the given node. In the next step the marker passer marks all nodes linked to the nodes marked in the previous step. Re-

peating this process, paths of any length between nodes can be found. Because a link in the knowledge base represents a relation (for example an IS-A or a PART-OF relation), a path, consisting of one or more links between two nodes, represents a semantic association between them. A simple example, given by Hirst, will illustrate how this can be used in lexical disambiguation.

Consider the sentence

Nadia's plane taxied to the terminal. (3.1)

This sentence contains three ambiguous words: *plane*, *taxi* and *terminal*. The NLU system will process the sentence from left to right, marking the frames representing each known meaning of each word. Consequently, the marker passer will find a path consisting of only one link between the frames airplane and airport-building, which were the starting points of *plane* and *terminal*. Another path of short length would be found between airport-building and aircraft-ground-travel. Aircraft-ground-travel was the starting point of *taxi*. These short paths indicate a strong semantic association, therefore the program can infer that air-plane, airport-building and aircraft-ground-travel correspond to the meanings of *plane*, *taxi*, and *terminal* in (3.1). If we put a certain limit on path lengths, then no paths will connect the frames representing the other meanings of the ambiguous words, namely wood-smoother, taxicab, and computer-terminal.

It will be clear that the marker passing system will find many paths that provide no useful disambiguation information. In the extreme case where we allow paths of arbitrary length, the system will find a path between each two nodes in the knowledge base (assuming that the knowledge base has no disconnected subgraphs). Apparently we must constrain the marker passing process.

In general the strength of the semantic association between nodes decreases with increasing path length. Therefore, we can assume that paths longer than a certain length n are irrelevant for the disambiguation process (n should be chosen to be small in comparison with the size of the knowledge base). Thus a useful constraint will be a limit on the path length. The disambiguation process can also be improved by restricting the types of paths. In Frail, the user can specify certain restrictions on the types of paths. For example, he can choose to pass markers only upwards in the IS-A hierarchy and not downward, so markers are passed to more general concepts. Restrictions that are the most useful should be determined experimentally.

Using these constraints, the marker passer might still find more than one path from a frame representing the meaning of a word to frames representing the different meaning of an ambiguous word. In this case we need a way to compare the strength of the semantic association represented by the different paths. Hirst uses the following heuristic rules:

- The shorter the path, the stronger the path.
- The more links that leave a node, the weaker the connections through that node.

Hirst points out that these methods (the constraints and the heuristic rules) are needed in the current version of the Polaroid Words, but are unsatisfactory because they rely on *magic numbers* (for example the maximum path length). As a possible improvement, he proposes some extra features as path strength and the weakening of activation when it gets further from the origin. The weakening of activation could be implemented by assigning weights to the links between the nodes. Interestingly, this resembles the type of activation spreading in connectionist systems.

3.2.3 Connectionism

Waltz and Pollack (1984; Pollack and Waltz 1982) and Cottrell and Small (1983; Small, Cottrell, and Shastri 1982) give connectionist models for NLU. Both models use the continuous deterministic scheme (section 2.3.3) and handle disambiguation problems.

Cottrell and Small use a representation scheme similar to the one used by Schank (1975) in his work on conceptual dependency. Syntax plays only a minor role in their scheme. In the work of Pollack and Waltz the syntax plays a more important role. Their system, for example, resolves syntactical ambiguity by using semantic knowledge. We will now give a more detailed description of these models.

In the model by Small et al. one can distinguish three levels. The first level is called the lexical level. This is the input level of the system; incoming words will activate the associated units. The second level is called the word sense level. This level gives the different senses of the words at the lexical level. The last level is called the case logic level. This level expresses the possible relationships between the predicates and objects presented at the second level. An example network will clarify their approach.

Consider the following input sentence:

A man threw up a ball. (3.2)

Figure 3.1a gives the state of the network, given by Small et al., after receiving

a man threw as an input. (The actual values of the weights on the excitatory and inhibitory connections were not given.) The number of plus signs represents the level of activation. A unit is called active if it has one or more plus signs. (The level of activation is a continuous value, so, for example, one plus sign refers to a value between c and $2c$, where c is a positive number.) One should note that the links are symmetrical, so activation can flow in either direction between two linked units.

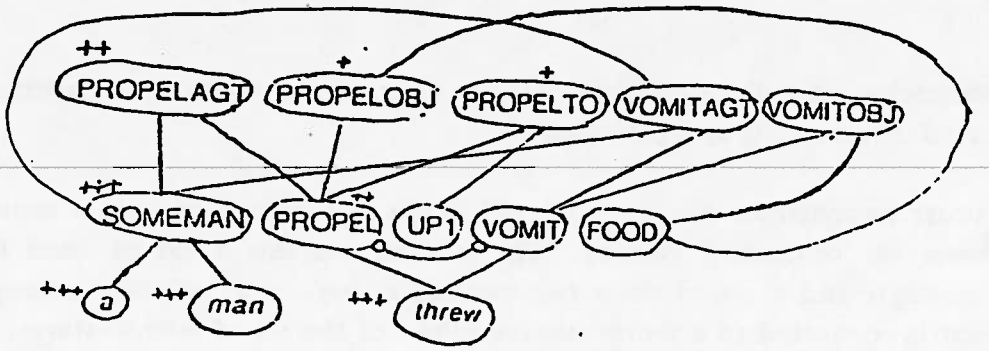
The phrase *a man* activates the units **a** and **man**. These units activate the unit **someman**. The article *a* will also excite, for example, **somewoman** (not shown in figure 3.1a), but an inhibitory link between **someman** and **somewoman** prevents them from being both active (we assume a person can't be both).

The unit **threw** excites **propel** and **vomit**, but as we will see in the next diagram the **vomit** unit needs more input to become active Thus only the unit **propel** becomes active. Although not explicitly shown in the figure 3.1a, the weight on the link between **threw** and **vomit** is smaller than that of the link between **threw** and **propel**, this represents the fact that *a man threw* is commonly associated with throwing an object. (Small et al. use a special type of link between **threw** and **vomit**, but simply decreasing the weight of that link suffices.) Moreover, the inhibitory link between **propel** and **vomit** will prevent them from both being active. The units **someman** and **propel** will activate **propelagt**.

The pattern of activity will change drastically when the word *up* is given as the next input, figure 3.1b shows this new pattern. The units **threw** and **up** activate **vomit**. Now, the unit **vomit** will become more active than **propel**, and because of the inhibitory link the activation of **propel** will decrease. The system will now settle in a new stable state, representing the fact that the phrase *threw up* mostly refers to vomiting.

Finally, the system will receive the input *a ball*, which reinforces the unit **propelobj** and inhibits the analogous **vomitobject**. The system will now settle in the stable pattern shown in figure 3.1c. This pattern represents the correct interpretation of (3.2). (Links are symmetrical; thus, **someball** excites **propelobj**, which excites **propel**.)

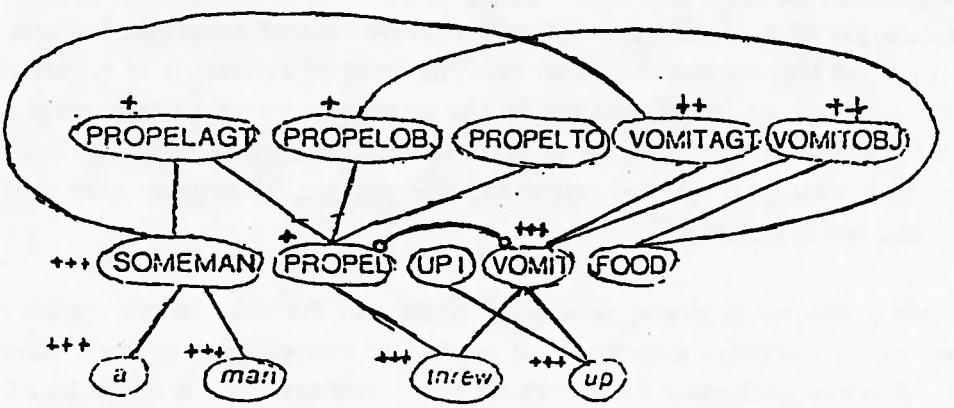
In the model of Waltz and Pollack (1984) we can distinguish four levels: an input level, where the network receives its input data; a syntactic level, that represents the syntactic parse of the input data; a lexical level, that represents the different senses of the incoming words; and a contextual level, that represents the context in which the sentence should be interpreted. This network is not part of some large network that processes an arbitrary input sentence, but has to be constructed for each specific input sentence. The sentence is run through a conventional chart-parser to construct the syntactic layer of the network; the lexical layer is constructed from all possible senses of the words, and the contextual layer is constructed from recent con-



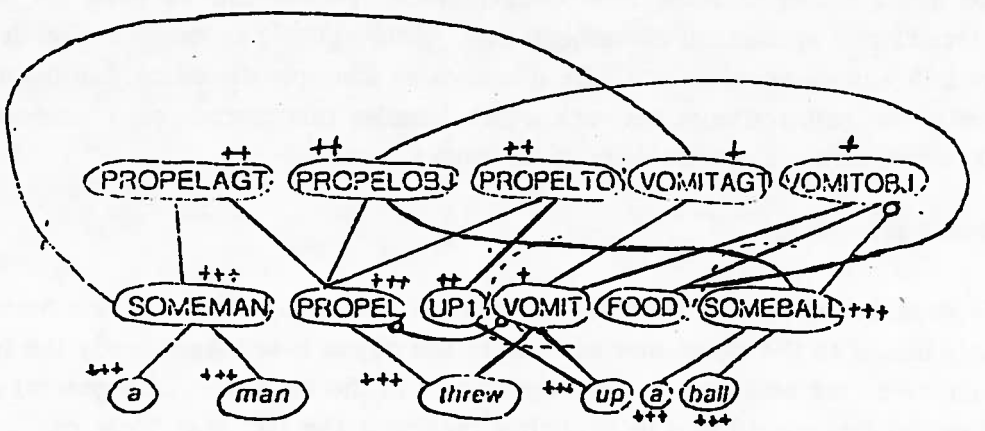
an inhibitory link

an excitatory link

(a) ***a man threw***



(b) ***a man threw up***



(c) ***a man threw up a ball***

Figure 3.1 The responses for different inputs of an example network by Small et al. (1982).

textual information, as will be discussed below. (The way in which these layers are interconnected is reasonably straightforward.)

In order to construct the context layer, Waltz and Pollack propose a connectionist scheme for contextual priming. The scheme contains a set of units that represent concepts and a set of units representing *microfeatures* of these concepts. Each concept is connected to a representative subset of the set of microfeatures; for example 'weekend' will be connected to 'day' and 'week', but also to 'restaurant', 'bar', and so forth.[1] Each concept will have at least one microfeature in common with another concept, and most concepts share many microfeatures with other concepts. Therefore, in general, all units will be indirectly connected to one another. In such a scheme the activation of a specific concept will activate related concepts, because the activation will spread throughout the network. The level of activation of a particular unit represents the strength of its relation to the priming concept. The current context is given by the state of the network at that moment, when new sentences come in and therefore new concepts will be activated, this pattern of activity may change, representing a change in context.

The main problems in the approach of Waltz and Pollack are the choice of a representative set of microfeatures and the setting of the weights in their scheme. Results obtained in psycholinguistic research on free word association might be of use when choosing the right set of weights. These results suggest that such a scheme requires the use of asymmetric connections; for example Deese (1961) finds that 'yellow' strongly primes 'color', however 'color' hardly primes 'yellow', but instead strongly primes 'blue'.

The given examples show how connectionist systems can be used for some forms of lexical and syntactical disambiguation. Reilly (1984) proposes a similar approach towards anaphora resolution. He discusses an example discourse, but does not give explicitly the connectionist network which handles this discourse. Therefore, it is difficult to evaluate the applicability of his model.

3.3 Rule-based processing

We consider again the example network given in figure 3.1a. In this network 'Bob' is only linked to the agent case and not to the object case. Apparently the functional structure of the sentence is preprogrammed in the network. In a general connectionist model one would like to explicitly represent the fact that 'Bob' can be ei-

---

[1] Interestingly, the use of a set of microfeatures is closely related to the distributed representation of concepts (section 2.3.2). In fact, the set of microfeatures that characterizes a certain concept can be viewed as a distributed representation of that concept.

ther an agent or an object; and represent a rule that specifies which of the two roles 'Bob' plays in a particular input sentence, based on the functional structure of the sentence.

In the model by Waltz and Pollack the syntactic layer of the network is constructed from the input sentence using a conventional chart parser. Also in this case, a more general model would require an explicit encoding of the grammar rules in the network, so that the same network would be able to handle a large number of input sentences. The problem we encounter here is a general one in connectionist schemes, namely, how to handle forms of processing that seem to require sets of rules.

Riley and Smolensky (1984) propose a connectionist model for rule-based processing in the domain of problem solving (the analysis of simple electronic circuits). As the most serious disadvantage of their approach, they mention the difficulty of performing symbol manipulation. Because of that, their model is capable of analyzing only one specific electronic circuit. And even for such a specific task they had to implement Ohm's law for each component of the circuit, i.e. there is a sub-network representing Ohm's law (or more precisely, the changes in voltage V, current I, and resistance R that are in agreement with Ohm's law) for each component in the circuit. Ideally one wants a system with Ohm's law stored only once and the ability to apply this law to a component in the system whenever that is necessary for the analysis of the circuit. This is achieved easily in a symbolic processing system where one stores a law in a general form, e.g. $V_x = I_x R_x$ where x is refers to any one of the components, but becomes difficult in connectionist models.

Instead of representing all possible instances of rules, like Riley and Smolensky, we propose to represent concepts only once in our scheme and use intermediate computing units to represent the possible bindings, given by the rules, between them. So, for example, instead of connecting 'Bob' to only the agent role, as is shown in figure 3.1, we link 'Bob' to both the agent and the object roles using some intermediate units. These intermediate units, functioning as binders, are linked to another part of the network that handles the syntactic parse of the sentence. The units representing the parse tree of the input sentence, and therefore the functional structure, will activate one of the intermediate units and thus indicate what role 'Bob' plays in the particular input sentence. We will discuss the implementation of a set of rules in detail in chapter 5, where we describe the representation of grammar rules in a connectionist network.

Finally we would like to mention that certain forms of rule-based processing in natural language understanding, such as logical inference to resolve complicated anaphora, are actually processed in a sequential manner, and therefore one should not model this processing in a parallel scheme. As a matter of fact, we find it question-

able whether the form of rule-based processing used in the analyses of electrical circuits is of a true parallel nature. A similar question arises with respect to the work on connectionist systems for logical inference by Ballard and Hayes (1984).

3.4 An integrated approach

In research on NLU systems there has always been much discussion as to what extent different forms of processing should be done in a parallel, integrated manner. Most conventional NLU systems follow a model where syntactic processing functions as a front end of the system. Thus, syntactic and semantic processing are strictly separated. Charniak (1983) proposes to add a marker-passing component which runs in parallel with the syntactic and semantic component of the NLU system. The marker passing facilitates disambiguation in the system and results in a more intergrated form of processing; in general it seems that disambiguation tasks in particular require a high degree of integration between the different forms of processing.

An important claim made in the work on connectionist models for NLU is that these models process in a completely integrated manner different sources of knowledge. However, we saw that previously developed models only handle a small part of the processing that is needed in NLU. We believe that these systems can only become of practical interest for NLU if they are also able to handle rule-based processing in a general way. Examples of rules, relevant in natural language processing are: grammar rules, rules for case filling, rules for definite noun phrase resolution, etc.

Therefore, we propose a connectionist NLU scheme which incorporates rule-based processing by using intermediate computing units that function as binders (as discussed previously) and handles different sources of knowledge in an integrated manner similar to that in previously proposed NLU schemes.

A complete NLU system has the ability to store the meaning of the incoming sentences in a knowledge base. The stored knowledge can be used in the processing of new input. Our proposed connectionist scheme does not have this ability, because, up to now, no efficient mechanism for direct storage and retrieval of knowledge in a connectionist model has been developed.

The design of a network that handles both disambiguation and rule-based processing would require many ad hoc decisions on the topology and the setting of weights and thresholds of the network. Therefore, we decided to focus in this research on rule-based processing. We choose to model parsing based on some context-free grammar rules. We will discuss the results of this work in the next chapter.

CHAPTER 4

A connectionist parsing system

4.1 Introduction

In this chapter we discuss a connectionist model for computing the syntactic parse of a sentence. The model is based upon a context-free grammar. This is not essential for our approach. The choice of another grammar, for example a transformational grammar, would complicate the design of the connectionist network, but would not require changes in the basic principles underlying our model. Our major concern has been to to demonstrate that rule-based, traditionally sequential, symbolic processing can be done in a highly parallel fashion, using a connectionist scheme. In this introduction we will discuss some general features of the model and motivate several design choices.

The syntactic categories of the grammar are represented in a localist manner, that is each syntactic category is represented by a unit in the network. We use *binder units* (explained later) to represent the different possible links between syntactic categories in a parse tree. This localist approach allows us to represent the grammar rules in a very straightforward manner and consequently determines the set of non-zero weights (giving the topology of the network), as we will see in the next section.

The grammar rules and the chosen representation determine how the units are interconnected. To set the weights on these connections we have to consider which updating function (or computational scheme) we are going to use. We choose the stochastic Boltzmann scheme (with a slight variation, namely output values of -1 and $+1$ instead of 0 and $+1$, section 5.3) over a deterministic scheme on the following grounds:

- **Local minima.** Using a deterministic scheme, the network might end up in only a local minimum. There are two approaches towards resolving this problem. One approach is to introduce some noise in the system as is done in the ISCON simulator (Small et al. 1983). This is basically equivalent to the simple random sampling

Monte Carlo method (Binder 1978) and is known to be inefficient in many cases as an optimization method. Another approach is to tune the set of weights in such a way that the system settles directly in the desired state given a certain input. This method requires many simulations for different inputs, because the behavior of the system is difficult to predict. These considerations lead us to the computational scheme of the Boltzmann machine. The Monte Carlo algorithm used in this scheme, introduced by Metropolis et al. (1953), is known to be much more efficient than the simple algorithm mentioned above. Also the chance of ending up in a local minimum is considerably smaller (theoretically this chance is zero, but only for an infinite number of computing steps). Moreover, as we argue below, the setting of the weights is easier than in the stochastic scheme.

• **Formal basis.** The Boltzmann machine formalism allows an interpretation of the weight between two units in terms of the conditional probabilities (section 5.3) and the chance of occurrence of a state is directly related to its energy. These properties are useful in the analysis of a network and facilitate the setting of the weights. (The deterministic model lacks such properties.)

4.2 The system

4.2.1 Topology

We distinguish two layers in the parsing system. The *input layer* consists of a number of computing units representing the terminal symbols of the grammar. An input sentence will activate some subset of these units. Connected to this input layer is a network that represents the parse trees of all non-terminal strings in the language whose length is not greater than the number of units in the input layer. This network, called the *parsing layer*, is constructed from *connectionist primitives*, which represent the context-free grammar rules. Figure 4.1 gives two examples of such primitives. The activation of all units of a primitive corresponds to the use of the associated grammar rule in the parse. The number of units in the parsing layer depends on the particular context-free grammar rules and the number of input units. The different parse trees are not represented by completely disjoint sets of units, but share common substructures. This will keep the size of the network manageable.

We use intermediate computing units to link the primitives together. These units play the role of binders in the network and, are therefore called *binder units*. The computing units representing the terminals and variables of the grammar are called *main units*.

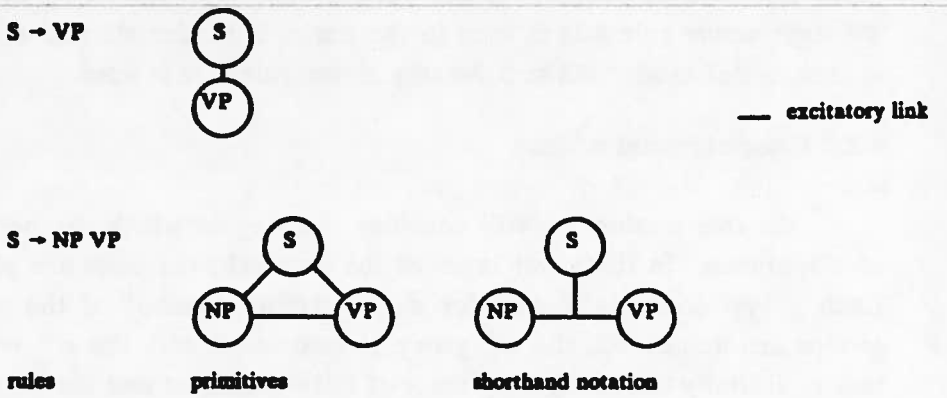


Figure 4.1 Some examples of connectionist primitives and their associated grammar rules. These primitives are the building blocks of the parsing network.

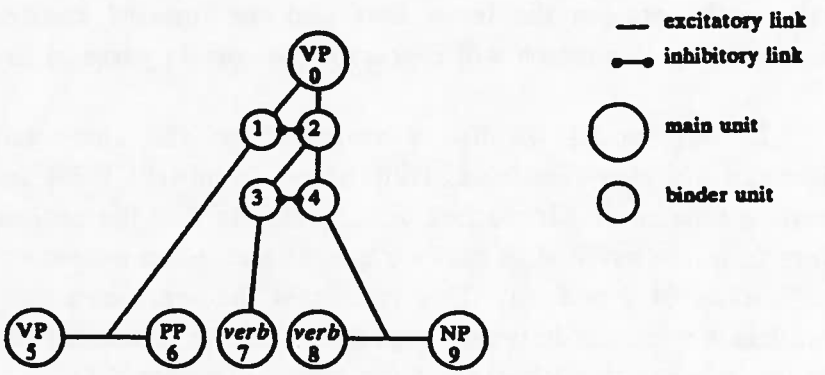


Figure 4.2 An example of the use of binder units (units 1, 2, 3 and 4). This network represents the grammar rules given in (4.1).

Figure 4.2 gives an example of the use of binder units. The four binder units are used to represent the fact that the main unit #0 is part of three grammar rules:

$$VP \rightarrow VP\ PP \qquad (4.1a)$$
$$VP \rightarrow verb \qquad (4.1b)$$
$$VP \rightarrow verb\ NP \qquad (4.1c)$$

The binders are linked in such a way, using inhibitory and excitatory connections, that

when the network reaches a global energy minimum the active binders (output equals +1) tell us which one of the three possible grammar rules is used in the parse of the input sentence to decompose the verb phrase represented by unit #0. So, if binder #1 stays active rule 4.1a is used in the parse, if binder #2 and #3 stay active rule 4.1b is used and if binder #2 and #4 stay active rule 4.1c is used.

4.2.2 Computational scheme

In this section we will consider the way in which the network finds the parse of a sentence. In the input layer of the network, the units are placed in *input groups*. Each group contains a unit for each terminal symbol of the grammar. The input groups are numbered; the n^{th} group is associated with the n^{th} word in the input sentence. Initially the computing units of both the input and the parsing layer of the network are inactive (their output is −1). As a sentence comes in, each word of the sentence activates the computing unit(s) representing its associated syntactic category or categories. So the first word of the sentence activates one or more units (depending on the number of syntactic categories associated with the word) in input group #1, the second word one or more units in input unit #2, and so on. After receiving input data, the network starts the relaxation process. During this process, the outputs of the activated computation units in the input groups are fixed at +1, while the outputs of other units in the input layer are fixed at −1, so that the network can find the optimal match between the input data and the internal constraints representing the grammar rules; this match will represent the correct parse of the input.

In our model we use a variation on the computational scheme of the Boltzmann machine (Fahlman, Hinton and Sejnowski 1983) and apply the simulated annealing scheme of Kirkpatrick et al. (1983) to find the optimal match. Our scheme differs from the original in that we use −1 and +1 as output values of our computing units instead of 0 and +1. This facilitates the representation of symmetrical interdependency relations between hypotheses in the scheme; there exists a one-to-one mapping between this scheme and the original (section 5.3).

The fact that this scheme searches for a global energy minimum and that at equilibrium the relative probability of a particular state of the system is given by its energy enables us to formulate general rules for the setting of the weights on the connections and the thresholds of the computing units.

We compute the average value of the output of each unit at the different temperatures used in the annealing scheme. In an example given below, we will see how these average values will change during cooling of the system; finally, at a temperature just above the freezing point of the system, the units with outputs close to +1 will represent the parse of the sentence. To find the temperature just above the

freezing point of the network, we consider statistical data on the behavior of the network during simulated annealing.

4.2.3 The setting of weights and thresholds

The setting of weights and thresholds is probably the most difficult problem in the design of a connectionist scheme. The set of weights and thresholds represents the internal constraints and therefore the knowledge in the system. So far we have described how units are interconnected in our parsing scheme; that is the set of links with non-zero weights. In this section we will discuss what values should be chosen for the weights on these links; we will also discuss the setting of the thresholds of the computing units.

In the Boltzmann formalism, the behavior of the system during relaxation can be described as a search for a global minimum in the energy function given in equation 2.4. From this equation, it follows that the contribution of a unit, k, to the energy is given by

$$E_{loc,k} = (-1/2 \sum_j w_{kj} s_j + \theta_k)\, s_k \qquad (4.2)$$

From (4.2) it follows that we can calculate the contribution of one unit to the energy of the network from the output value of that unit and its nearest neighbors (that is those units with a link to that unit). Using equation 4.2 we can write for the total energy

$$E = \sum_k E_{loc,k} \qquad (4.3)$$

Given the fact that the network searches for a global energy minimum, we can, to a first approximation, analyze the behavior of the network by assuming that each unit and its direct neighbors will choose output values such that $E_{loc,k}$ becomes minimal. However this method gives only a rough approximation of the actual behavior, because minimizing E_{loc} for one particular unit often conflicts with minimizing E_{loc} of other units. To get a better insight in the behavior of the system we therefore consider the contribution to the global energy of small groups of units.[1] Because of the homogeneous structure of our network we only have to consider a limited number of cases.. From these considerations follows a set rules for setting weights and thresholds in our parsing scheme. These rules are given below.

---

[1] Of course, for an exact analysis one would have to consider all possible states of the (total) network; this becomes clearly infeasible for networks with more than about 25 nodes.

• **Excitatory links.** Symmetry considerations and the fact that there is no distinction between bottom-up and top-down parsing in the network lead directly to the choice of weights on the excitatory links. Figure 4.3a shows some excitatory links in a typical configuration. The network represents two grammar rules

$$VP \rightarrow verb \qquad (4.4a)$$
$$VP \rightarrow verb\ NP \quad (4.4b)$$

Rule 4.4a is represented by the units 0, 1, and 3; rule 4.4b by the units 0, 2, 4, and 5. During the relaxation process, our network has to decide between rule 4.4a and rule 4.4b or neither of them. There is no *a priori* preference of one rule over the other. Therefore, because unit #2 is connected to two other units representing the left-hand side of rule 4.4a and unit #1 is connected to only one unit representing the left-hand side of rule 4.4b, we have to make the weight on the link between units #1 and #3 twice as strong as the links between units #2 and #4 and between units #2 and #5. (This can be easily generalized for grammar rules with more symbols; one choses the weights such that the sum of the inputs at the binder units is equal for all possible grammar rules.) So we choose $w_{2,4}$ and $w_{2,5}$ equal to some positive constant α and we set $w_{1,3}$ to $2.0\,\alpha$. One should note that only the ratio of these numbers is of importance. The magnitude of these numbers is going to determine at what temperature in our simulated annealing scheme the system is going to freeze, but the temperature is only a formal parameter introduced to do simulated annealing and has no meaning for our final result.

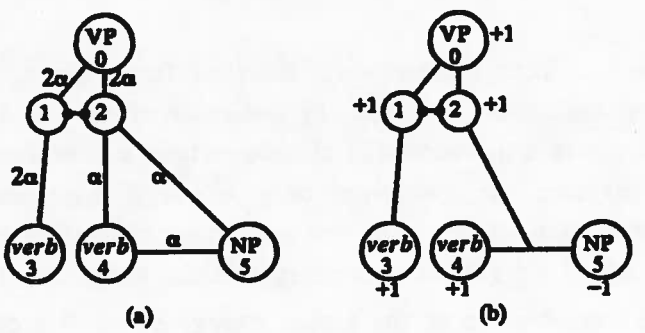


Figure 4.3 Part (a) shows some excitatory links and one inhibitory link in a typical configuration in the parsing network. Part (b) gives an example of a state of the network that we would like to prevent by making the inhibitory link stronger than the strongest excitatory link.

For the use of a grammar rule in the parse, the presence of each symbol in the rule is equally important, and therefore we connect the units in a connectionist primitive representing a grammar rule with links of equal strength, so $w_{4,5} = \alpha$. And finally, because bottom-up and top-down parsing is completely integrated and of equal importance in our networks we choose $w_{0,1} = w_{0,2} = 2.0\,\alpha$.

• **Inhibitory links.** Binder units are the only units with inhibitory connections. Each binder unit is connected by an inhibitory link to another binder unit. As we discussed previously, they are used by the network to choose between grammar rules in the parse of the input sentence. Therefore we must avoid states of the network where both units of a pair of binder units are active; these would be meaningless states of the network, because they would represent the simultaneous application of two grammar rules to decompose a syntactic category in the parsing process. Figure 4.3b gives an example of the network given in figure 4.3a in a state that we must avoid. A way to prevent this situation is by making the inhibition stronger than the single excitatory link between unit #0 and #2; exactly how strong is directly related to the threshold values of the binder units, as we will see below.

• **Thresholds.** So far we have implicitly assumed that all thresholds were chosen equal to 0. Because we use the $-1/+1$ model, this means that the links represent symmetric dependence relations between units (section 5.3). So the influence of a unit with output -1 on its surrounding is opposite to, but of the same magnitude as, the influence of that unit with output $+1$. This means in terms of energy that, for example, the energy of a connectionist primitive with all units on (output is $+1$) is the same as the energy of that primitive with all units off (output is -1); this follows from equation 4.3. This symmetry expresses the fact that there is no a priori knowledge in the system on whether a certain connectionist primitive (or grammar rule) should be used or not in the parse of an input sentence. However, with the binder units we do have a priori knowledge which tells us that the dependency relation of those units should be asymmetrical. This is most easily demonstrated with an example. In figure 4.4a we consider two states of a network with one main unit and its two binders. When the thresholds of the units is zero the energies of the states *A* and *B* are the same. However, we know a priori that only the state *A* can be part of a correct parse of a sentence; therefore we would like to give this state a lower energy than that of state *B*. This can be done by introducing a positive threshold in the binder units. This positive threshold gives a binder unit a preference for staying inactive and only becoming active if there is support from its neighbors 'above' and 'below' the unit. We don't introduce such a threshold in the main unit because we don't have a priori knowledge which tells us that the main unit is more likely not to be part of the parse than to be part of it.

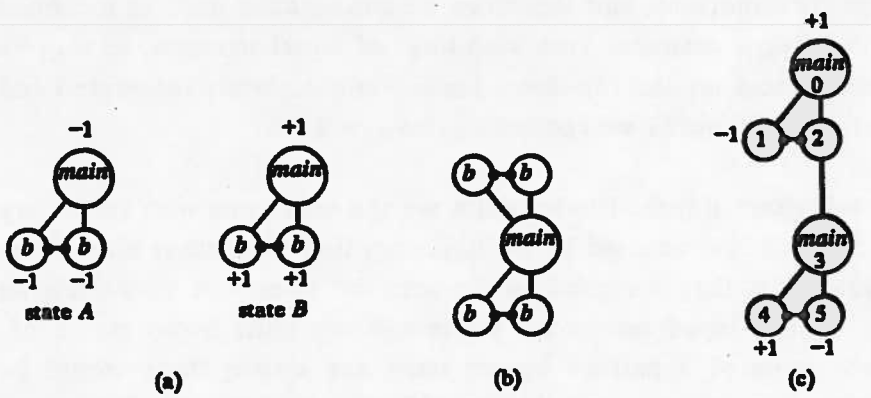


Figure 4.4 Part (a) gives two states which have equal energies in case the thresholds of the computing units are equal to zero. To give state A a preference over state B we introduce a positive threshold in the binder units (this results in $E_A < E_B$). Part (b) and (c) show main units in highly symmetrical configurations. We have to introduce a small negative threshold in such units.

To choose the size of this threshold we have to take into consideration the strength of the inhibitory connection between binders. The threshold and the weight on the inhibitory link should be such that without support from surrounding units both binders stay inactive; with support from active surrounding units one of the binders is likely to become active; and finally the activation of both binders simultaneously is extremely unlikely. From the energy of a pair of binder nodes we can deduce that setting the weight on the link between them to $-\beta$ (where β is a positive value; as explained above, this value should be larger than 2.0 α) and the threshold to $\beta - \alpha$. This will result in the required behavior.

Also, some of the main units need a non-zero threshold depending on their environment; see figures 4.4b, c. Figure 4.4b shows a main node surrounded by pairs of binder nodes. In this case, if one unit in each of the pairs of binder units is active, the main unit should be part of the parse and therefore be active. However, when the main unit has a threshold equal to 0, the energy of the network with the main unit on is equal to the energy of the state with main unit off. To remove this symmetry we have to choose a non-zero threshold, in this case one which favors the active state; we choose $\theta = -2\alpha$.

Finally we consider figure 4.4c. As we discussed above, a binder unit only becomes active if there is support for this unit coming from the unit above and the unit below the binder. Therefore, because the two binders #4 and #5 cancel each other's effect on main unit #3, both unit #3 and unit #2 will stay inactive when the threshold of unit #3 is zero. To solve this problem we give this unit a negative threshold (we choose -2α), which leads to a consistent state of the network with both units #1 and #2 active.

• **Summary.** Choosing $\alpha = 1.0$ and $\beta = 3.0$, the rules become as follows:

$weight_{excitatory\ link}$	+1.0	*in primitive with three units*	(4.5a)
	+2.0	*in primitive with two units*	(4.5b)
$weight_{inhibitory\ link}$	−3.0		(4.5c)
threshold	0.0	*main unit*	(4.5d)
	−2.0	*main unit in symmetrical environment*	(4.5e)
	+2.0	*binder unit.*	(4.5f)

A *main unit in a symmetrical environment* is a main unit only linked to pairs of binder units (that is connected to both binders) and at most one other binder unit.

Although the local analyses and symmetry considerations on which these rules are based won't guarantee the right global behavior, good simulation results of a network with weights set according to these rules show that apparently such a local analysis gives a reasonable estimate of the global behavior of the parsing network. This is most presumably a consequence of the highly homogeneous structure of our parsing scheme (the networks are built from a small number of primitives).

The setting of the threshold of the main unit in figure 4.4c is clearly the most specific case discussed in this section and is easily overlooked in the design of a network. We will therefore in section 4.4 consider what the consequences are if we choose a zero threshold for this unit.

Instead of using a local analysis to set the weights and thresholds, one might consider the use of a learning algorithm. So far, the only known learning algorithm is the one developed by Hinton and Sejnowski (1983a, 1983b; Hinton, Sejnowski and, Ackley 1984) for the Boltzmann machine (section 2.3.3). This algorithm has been successfully used in some small-scale examples. In these examples large training sets (several thousand samples) were used. It is not clear whether this algorithm can be used to set the weights and thresholds in our network; the main problem is to define an appropriate learning set.

An interesting alternative for the setting of weights and thresholds is a method that combines local analysis with a learning algorithm. That is, first a local analysis is used to find a rough estimate of the set of weights and threshold and subsequently one uses learning algorithm to 'tune' the network. This is probably what happens in the human brain, namely one is born with some inherited structures in the neural tissue and during life a learning process adjusts the strengths of connections between the neurons. (Feldman 1982 discusses changes in strenght of connections in neural networks.)

4.3 The design and testing of a network

To illustrate our model we will now consider an example. This network is based on the following context-free grammar rules:[1]

S	→ *NP VP*	*NP*	→ *determiner NP2*	
S	→ *VP*	*NP*	→ *NP2*	
VP	→ *verb*	*NP*	→ *NP PP*	(4.6)
VP	→ *verb NP*	*NP2*	→ *noun*	
VP	→ *VP PP*	*NP2*	→ *adjective NP2*	
PP	→ *preposition NP*			

We will represent five input groups. In a complete network each input group has a unit for all terminals of the grammar; however to make our example network less complex, we will not represent each terminal in each input group.

For the grammar rules in (4.6), we can construct connectionist primitives similar to those given in figure 4.1. To build the parsing layer upon the input layer, using these primitives, we consider the different possible ways in which the syntactic categories can be grouped according to the grammar rules, and design a network that represents those possibilities. One way we could have proceeded is by designing a set of networks, each representing the parse of one unique input sentence, linking all these networks to the input layer, and placing inhibitory connections between them. These inhibitory connections should guarantee that after the parsing network is given an input sentence, only the sub-network representing the parse of the input would remain active.

Apart from the question whether the design of such a network is even feasible, there are several reasons why we did not take this approach. Firstly, many parse trees have common sub-structures. So one can save computing units by representing such structures by one set of units and linking that structure, using binder units, to the different parse trees represented in the network. Secondly, in this model we try

---

[1] These rules are taken from an example in Winograd (1983).

to represent rule-based, symbolic processing in a connectionist scheme. We especially would like to preserve the merits of such processing as much as possible. The power of symbolic processing clearly comes from the fact that relatively few symbols can be ordered in a large number of ways, each representing a different overall meaning. In our scheme the symbols can be compared with the main units, and the binders units are used to 'order' them in many different ways. Therefore we strive to keep the number of main units small and use binders to be able to order them in many different ways. Finally, main units do represent general concepts, such as 'noun phrase'; it is more likely that in the human brain these concepts are represented by one structure[1] than by many different ones. This is another reason why one should try to minimize the number of main units. This should only be used as a guideline, because it remains to be seen whether the concepts we use in our scheme are representative for the concepts humans use to determine the structure of a sentence (Stabler 1983).

So, instead of constructing a network from a set of separate networks, each representing the parse of a sentence, we take an approach in which we try to share common syntactic structures between parse trees and minimize the number of main units. Following these guidelines we can construct from the input layer, using the connectionist primitives, a network like the one given in figure 4.5.[2] The weights and thresholds in this scheme are set according to (4.5).

To test our network we use the following 'sentences':

noun verb preposition determiner noun	(4.7*a*)
verb noun preposition determiner noun	(4.7*b*)
adjective noun verb determiner noun	(4.7*c*)
adjective noun verb	(4.7*d*)
noun {noun verb} preposition determiner noun	(4.7*e*)

The sentence *John ran down the hill* is an example of an input sentence that corresponds to (4.7a); (4.7e) gives a syntactically ambiguous input activating two units in input group #2 (*John watches from the hill* is an example of a corresponding input sentence). The parse trees of these sentences are given in figure 4.6 (except for sen-

[1] Whether this is a group of units as in the distributed representation or one unit as in the local representation is irrelevant in this context, because in both representations each concept has one unique representation; that is one specific pattern of activation or one particular unit.

[2] Some simple input sentences show that the multiple units for NP's, NP2's, and PP's are necessary; however, one could further minimize the number of VP's. However, this results in a network where the connectionist primitives are less visible, and which is therefore harder to understand.

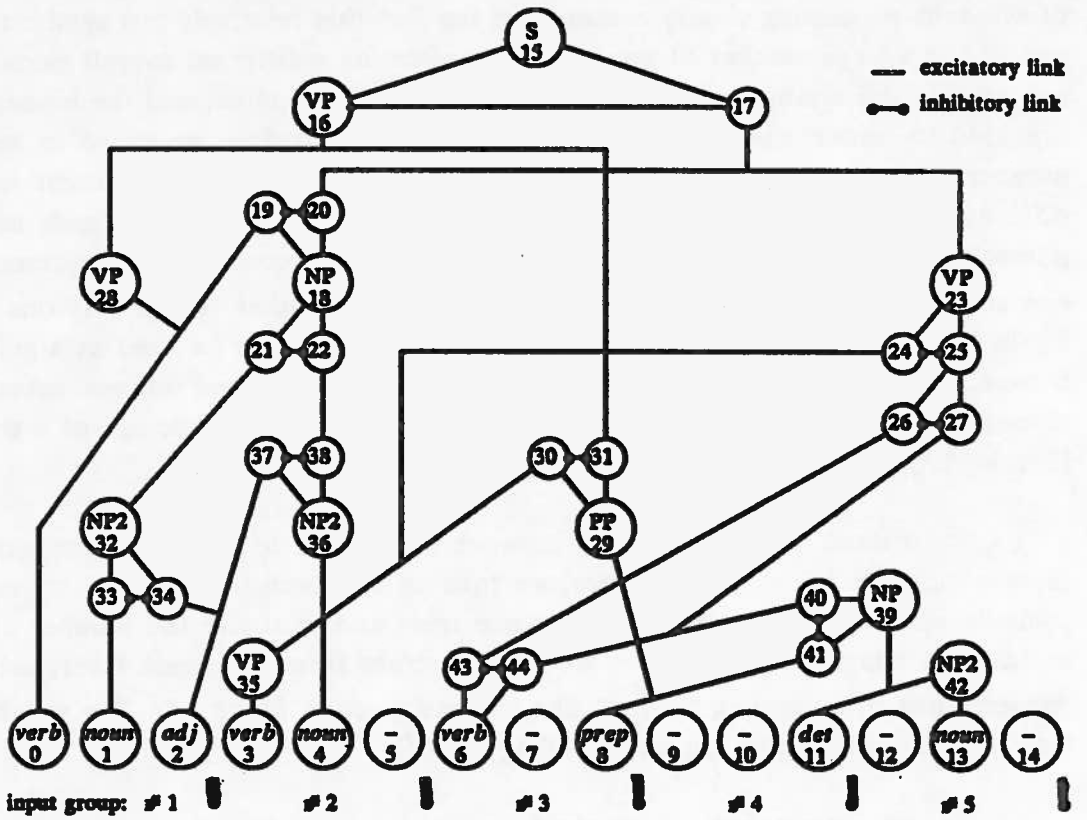


Figure 4.5 An example parsing network based on the grammar rules given in (4.5). Input group #1 consists of units 0, 1, and 2; group #2 consists of units 3, 4, and 5, and so forth.

tence (4.7e); its parse tree is equivalent to that of sentence (4.7a)). For each input sentence we ran a simulation of the parallel network on a serial machine using a simulated annealing scheme. To apply this scheme, one has to choose a descending sequence of temperatures such that the system has a reasonable chance of finding the state with the global energy minimum. Therefore one starts at a high temperature and first cools rapidly; once the system approaches the freezing point (the point at which it settles down in a state with a local or a global energy minimum; in this state the temperature is too low to escape from this minimum) one should cool very slowly. As we will see, we don't have to freeze the system completely; the right parse of the input sentence is found at a temperature just above freezing. At each temperature above the freezing point one has to take sufficient computation steps to allow the system to reach equilibrium at that temperature.

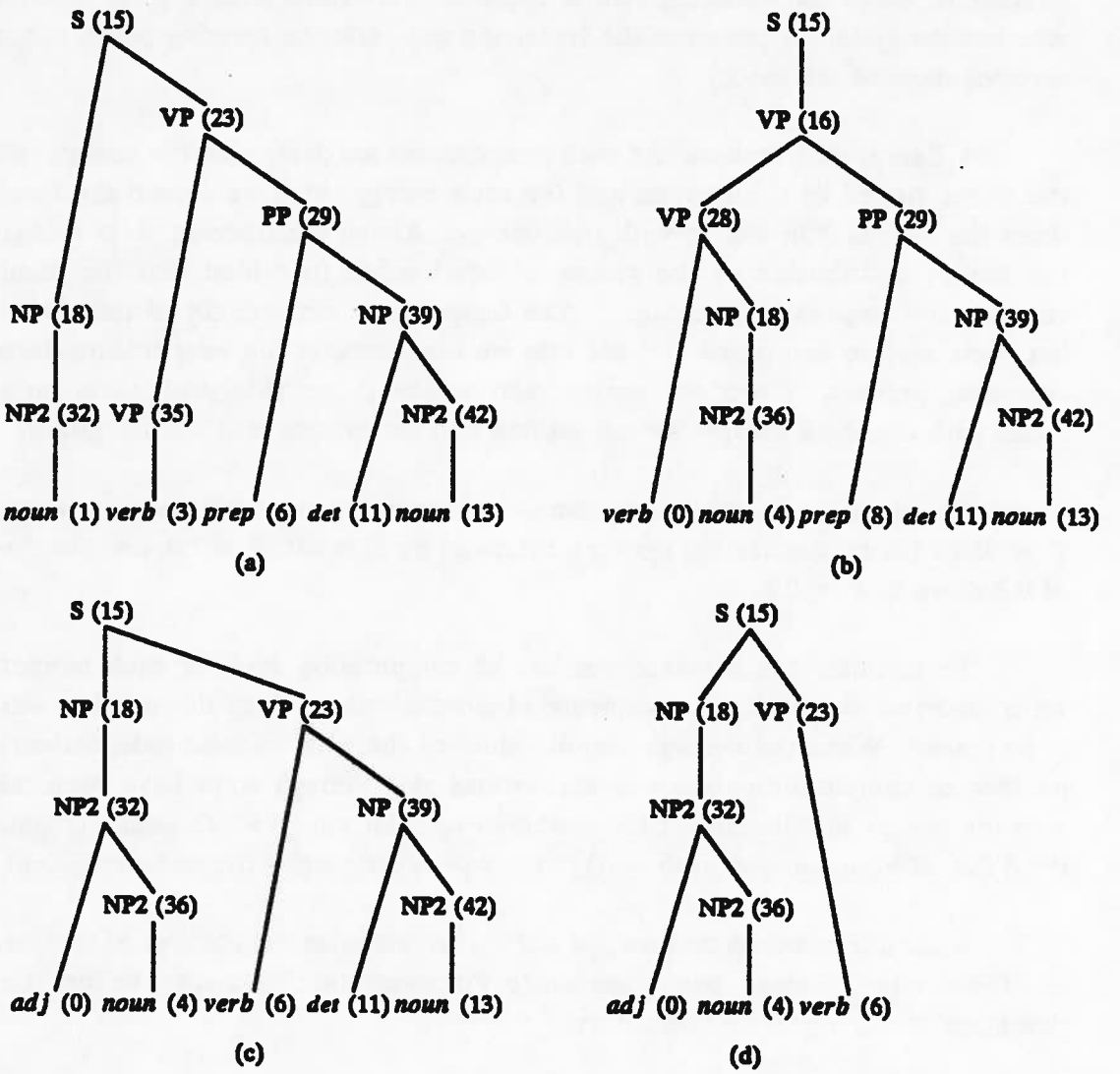


Figure 4.6 The parse trees of the sentences (4.7a) to (4.7d). The numbers between parentheses are the numbers of the corresponding computing units in the parsing network.

To be able to choose the sequence of temperatures and the number of computation steps we did some test runs with the network. The sequence of temperatures was determined by considering the following data accumulated during the annealing process:

• **Changes in output values.** At each temperature we determine for each unit the ratio between the number of times the output value of that unit changes and the number of times the updating rule is applied. This ratio gives a good indication of whether the system approaches the freezing point. (At the freezing point, the ratio is zero for most of the units.)

• **Energy distribution.** At each temperature we determine the energy values of the states visited by the system, and for each energy value we record the number of times the system is in a state with that energy. Above the freezing point this gives us the energy distribution of the system at equilibrium (provided that the number of computation steps is large enough). The temperature dependency of this distribution has been used to determine at what rate we can decrease the temperature during the annealing process. When the energy distribution shows that the system only visits states with the same energy, we can assume that the system is in a frozen state.

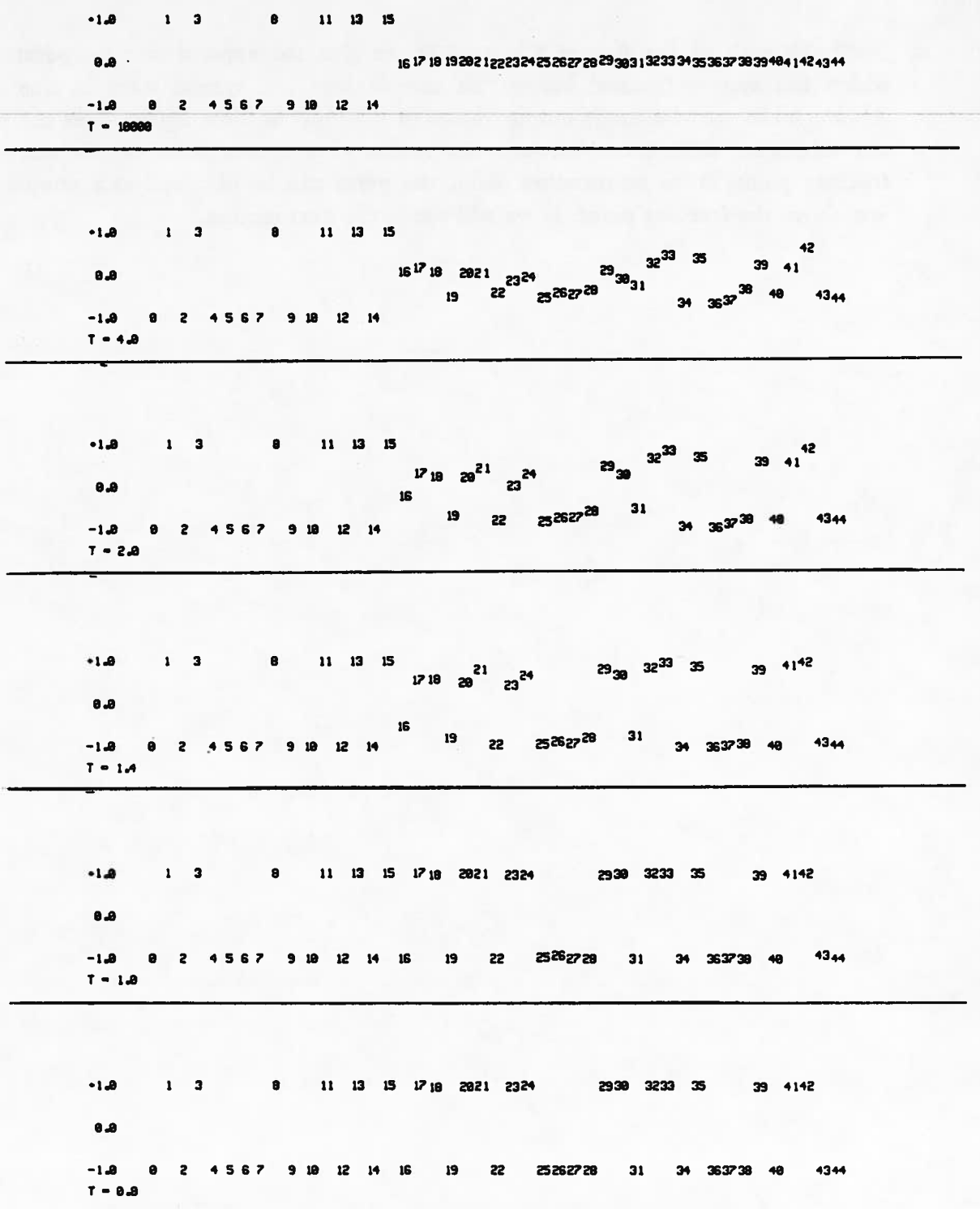
Based on these indicators, we choose a sequence of temperatures starting at $T = 10000$ (to randomize the system), followed by $T = 4.0$, $T = 2.0$, and then in steps of 0.2 down to $T = 0.2$.

To estimate the required number of computation steps at each temperature, we considered the results of a sequence of simulations in which this number was slowly increased. When the average output values of the units become independent of the number of computation steps one can assume that enough steps have been taken to scan the energy distribution of the system at equilibrium. Two thousand computation steps (i.e. 2000 updates of each unit) per temperature turned out to be sufficient.

It should be noted that we did not try to minimize the number of temperatures and the number of steps per temperature. For example, one could take less computation steps at the higher temperatures.

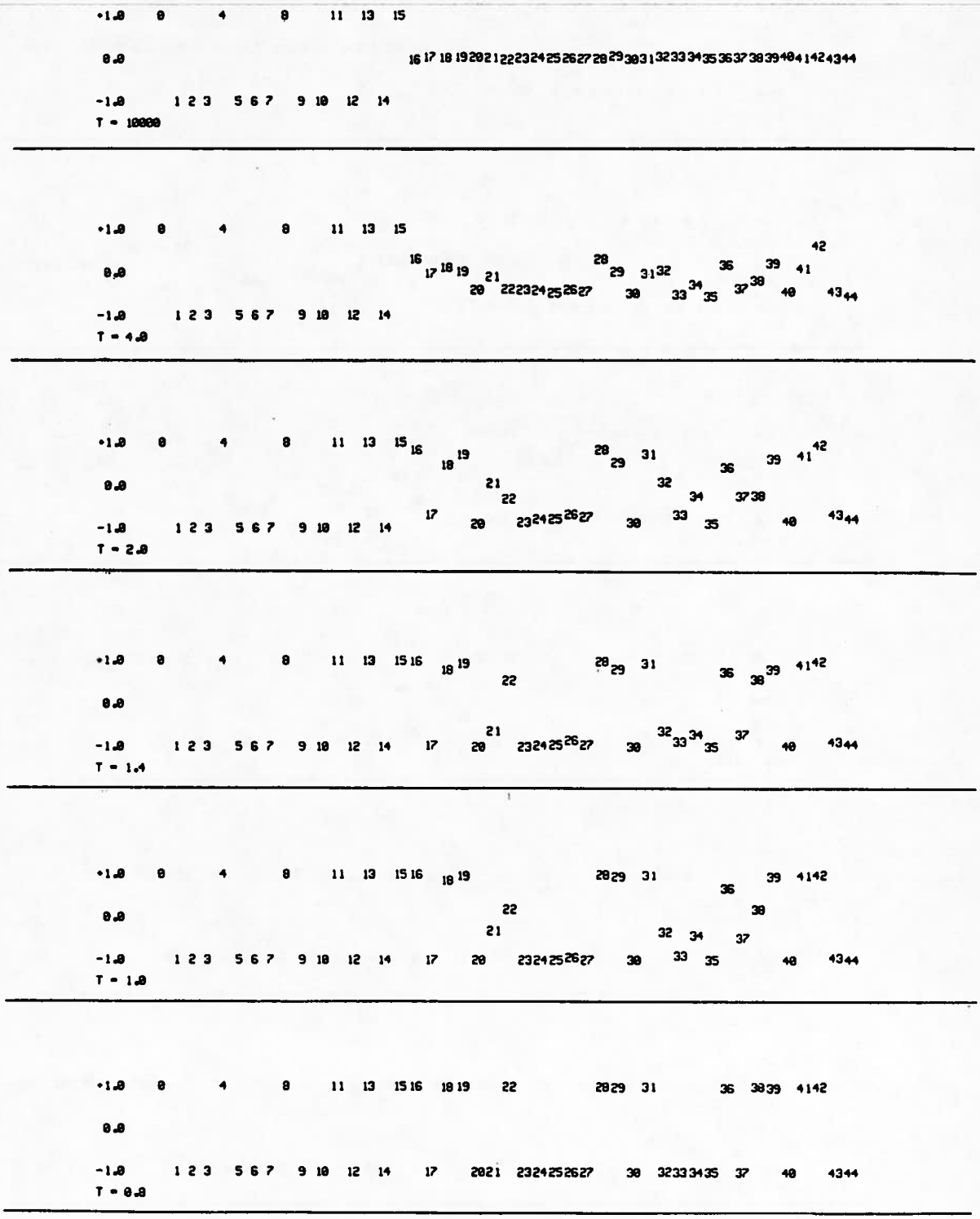
Figures 4.7a to 4.7e give the results of the annealing process for the input sentences 4.7a to 4.7e; we give six temperatures. In these figures, each panel shows the average output value of each computing unit at the temperature given below the panel. The numbers 0 to 44 are the numbers of the computing units; the vertical position indicates their average output value on the interval [−1,+1]. Comparison of these results with the parse tree of this sentence given in figure 4.6 shows that the time average of the outputs of the units, at a low non-zero temperature, corresponds to the correct parse of the input sentence if one chooses the units with outputs close to +1 as being part of the parse tree. At low temperatures there is a clear distinction between units with output close to +1 and the other units, as can be seen in the figures 4.7a to 4.7e.

In each of the figures 4.7a to 4.7e we give the approximate temperature at which the system freezes; below this temperature the system stays in one state. Although the set of average output values of the units in these figures does not reveal any significant differences between the system in a frozen state or just above the freezing point; more information about the parse can be obtained at a temperature just above the freezing point, as we will see in the next section.

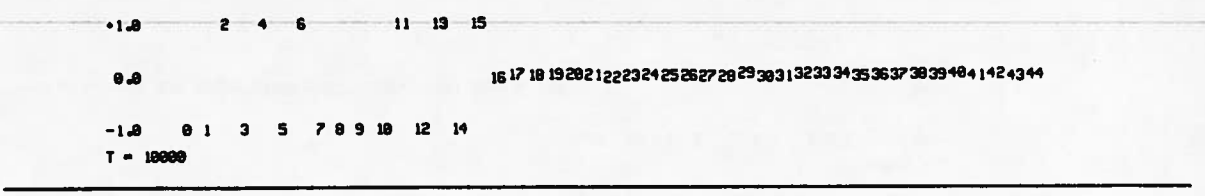


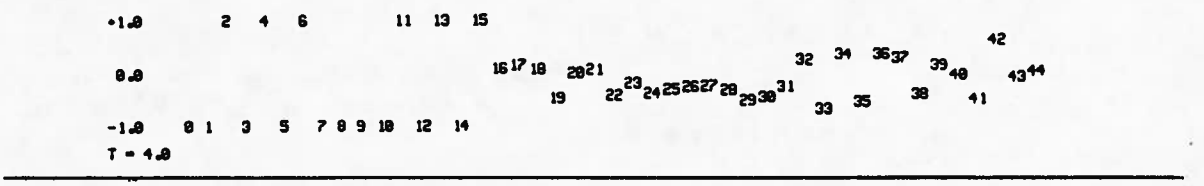
(a) Input sentence (4.7a); the system freezes at approximately $T = 0.8$.

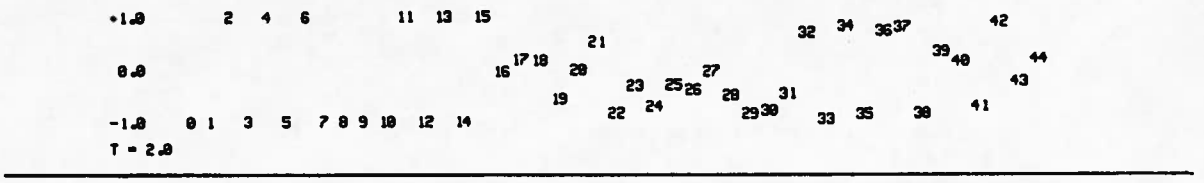
Figure 4.7 The results of parsing the sentences (4.7a) to (4.7c) with the network given in figure 4.5. Represented are the average output values (along the y-axis) of each unit at different temperatures used in the simulated annealing scheme. The numbers in the graph correspond to the numbers of the computing units in the parsing network. During the simulations, the outputs of units 1 to 15 were fixed. (Units 1 to 14 represent the input; unit 15 gives some top-down priming. Fixing unit 15 does speed up the simulated annealing process, but is not necessary for finding the correct parse.)

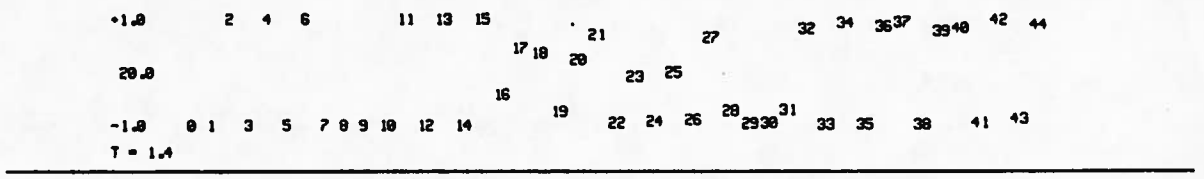


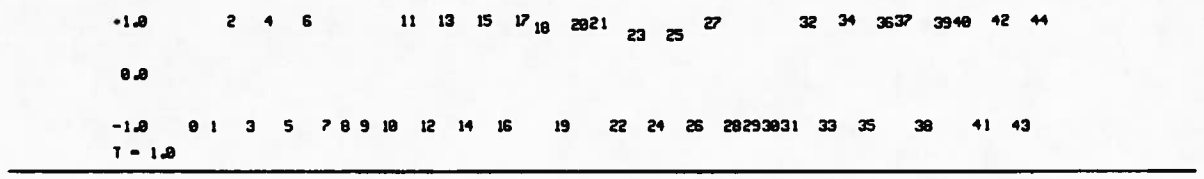
(b) Input sentence (4.7b); the system freezes at approximately $T = 0.8$.

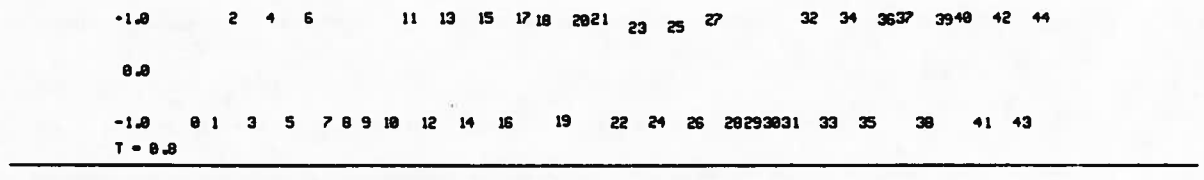




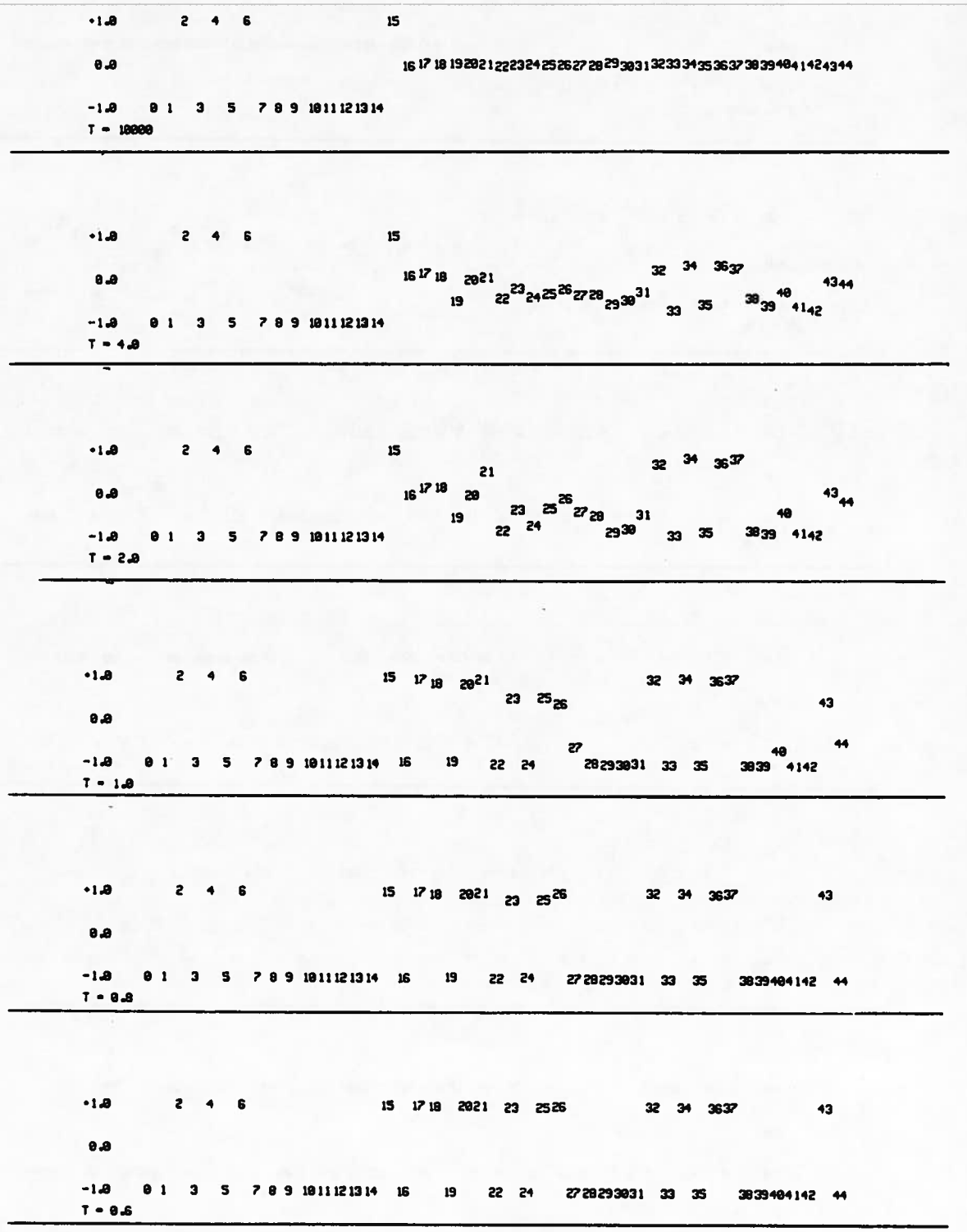




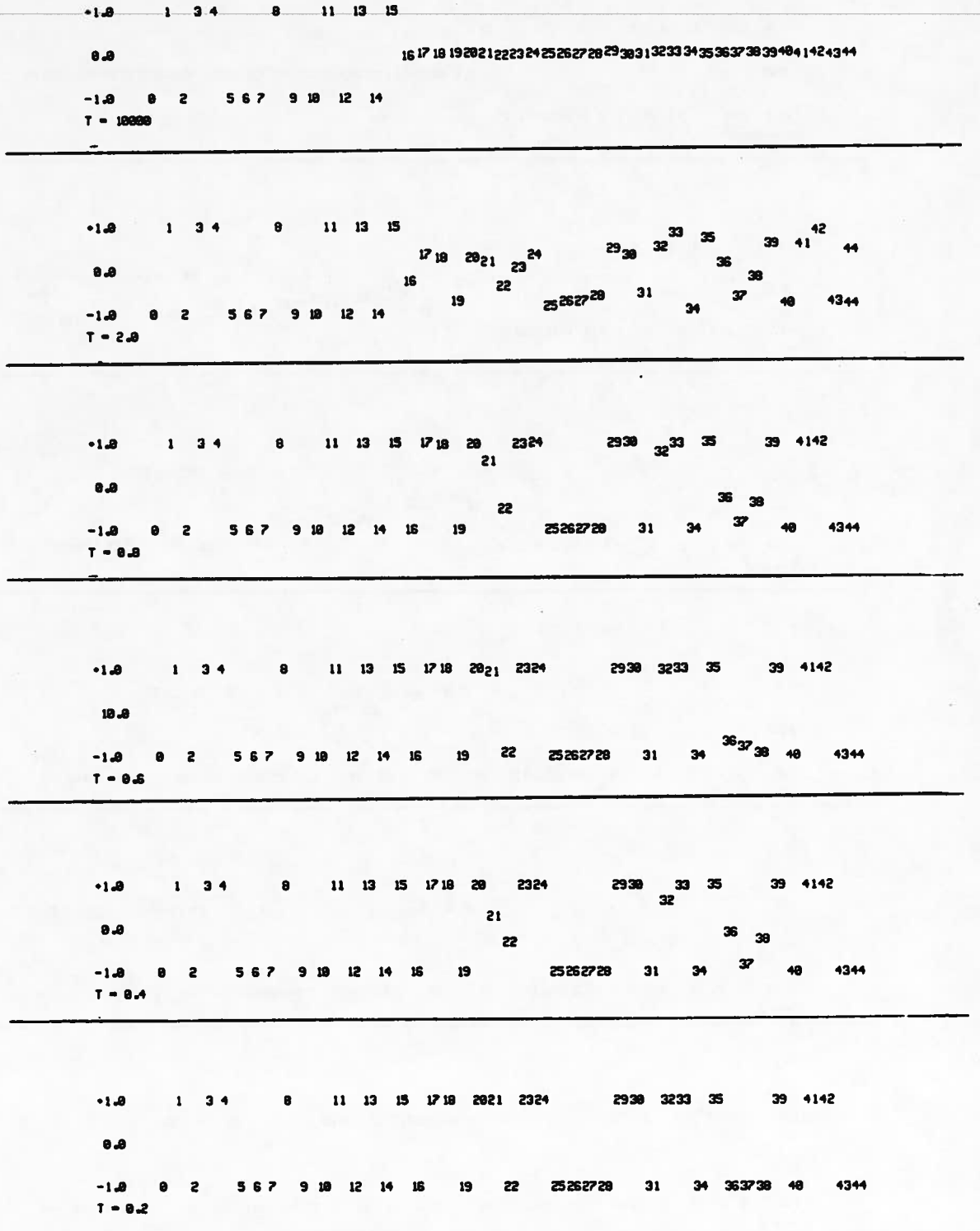




(c) Input sentence (4.7c); the system freezes at approximately $T = 0.8$.



(d) Input sentence (4.7d); the system freezes at approximately $T = 0.6$.



(c) Input sentence (4.7c); the system freezes at approximately $T = 0.2$.

4.4 Changing weights and thresholds

The weights in the example network given above were set in accordance with the rules given in (4.5). Because the setting of the weights and thresholds is an important issue in connectionist models, we will now consider what happens if we change some of them. We use the example network given in figure 4.5.

First we set the threshold of main unit #32 equal to zero; originally this threshold was set to −2.0, following rule (4.5f). The simulation results show that in the frozen state the system gives the correct parse, except for the main nodes #18 and #32 and the binder #21; that is the average output values of those units are −1.0. However the average of the output values of these units at a temperature just above freezing is 0.0. So at that temperature these units are part of the parse of the sentence for 50% of the time (all other average output values are close to −1.0 and +1.0, consistent with the correct parse). This result can be explained as follows. Just above the freezing point the system jumps between two states, namely:

- state **a**, all units of the correct parse are active; and
- state **b**, same as state **a**, except for the units 18, 21, and 32.

States **a** and **b** have approximately the same low energy; however to jump between these states the system has to visit a state with a higher energy. At a temperature above the freezing temperature there is enough thermal energy to visit the intermediate state with a higher energy; in other words the system has a reasonable chance to visit in the intermediate state compared to the chance to be in the lower energy states **a**, and **b**. Therefore, the system jumps between state **a**, and **b** and the average output value of the units 18, 21, and 32 will be around zero. However, when the temperature is lowered the system freezes, that is it will settle in one of the states **a** or **b**, and there is not enough thermal energy to jump to the other state with minimal energy.

This example clearly demonstrates that the average output values of the units give more relevant information when determined just above the freezing point of the system than at or below that point. It also demonstrates how the Boltzmann mechanism not only finds a global minimum in the energy, but just above the freezing point the system jumps between a number of states with energies close or equal to the global energy minimum. This is an important advantage over a deterministic scheme. Even in case such a scheme manages to find one of the states **a** or **b**, it is very unlikely that a deterministic scheme, just before finding **a** or **b**, would pass through the other state with minimal energy. This example also shows that if we don't follow all the rules given in (4.5), the system does not behave as well as when we do; however the model still comes up with a result close to the correct parse.

We will now consider what happens if we increase both the strength of the inhibitory links between binder units and the thresholds of these units. We choose a weight of −20.0 on the inhibitory links and thresholds of +19.0. These values are in accordance with the general rule for setting the thresholds on binder units and the weights on inhibitory links between them (section 4.2.3). In this case we don't find any significant differences between the simulation results with this new choice of weights and thresholds and those using the original values.

This is an interesting result, because with this choice of weights it becomes extremely unlikely, at low temperatures, to find a pair of binder units with both outputs equal to +1.0. Such a pair would give a large positive contribution to the energy; see equation (4.2). Therefore, the pairs are functioning as three-state switches with at most one unit with output equal to +1.0. This is useful during the search for a correct parse (or global energy minimum), because a pair with both outputs equal to +1 corresponds to the obviously incorrect situation in which two grammar rules are applied at the same time to decompose a syntactic category.

In figure 4.2 we saw two pairs of binder units linked in such a way that they can choose the application of one specific grammar rule out of three. Using a similar approach one can design a network from pairs of binder units that can select one rule out of a large collection; such networks will be useful in general connectionist schemes for rule-based processing.

CHAPTER 5

Modifications of the Boltzmann scheme

5.1 Introduction

The proper functioning of a connectionist model using the Boltzmann formalism depends on the topology of the energy surface of the network. Firstly, this energy surface should, ideally, have an unique global minimum that represents the optimal match between input data and the constraints in the network. Secondly, this surface should be smoothly curved in such a way that the simulated annealing scheme easily finds the global minimum. Studies of physical systems (Binder 1982) show that the efficiency of the simulated annealing scheme strongly depends on the form of the energy surface. For example, if the surface contains two steep minima, a global one and a local one, separated by a high energy barrier, then during the simulated annealing scheme the system might end up in the local energy minimum, from which it has only a very small chance to escape at low temperatures. In such cases the annealing scheme becomes very inefficient.

In section 5.2 we will address the question of how we can set weights and thresholds in a network such that the system will have a unique global minimum. However, we do not study the actual form of the energy surface in detail; this should be a topic of future research (section 6.2).

In section 5.3, we will discuss the use of -1 instead of 0 as an output value of the computing units. This modification is of practical interest, because it facilitates the setting of thresholds in our parsing network.

5.2 An alternative energy function

Equation (4.3), repeated here as (5.1), gives the energy as a sum over the contributions of individual units.

$$E = \sum_k E_{loc,k} \qquad (5.1)$$

From equation (4.2), repeated here as (5.2), it follows that the contribution to the energy of an individual unit, E_{loc}, is given by the value of the output of that unit and its neighbors.

$$E_{loc,k} = (-1/2 \sum_j w_{kj} s_j + \theta_k)\, s_k \qquad (5.2)$$

As discussed in section 4.2.3 we can estimate the global minimum in the energy by assuming that the surrounding of each unit is such that E_{loc} is minimal for that unit; this will give only a rough estimate of the global energy minimum, because the minimization of a contribution to the energy by a specific unit will often conflict with the minimization of the contributions by its neighbors. We proceeded in section 4.3.2 by considering the energy of small groups of units to get a better estimate of the global energy minimum. However, an exact value of the global minimum in the energy given by (5.1) and the corresponding states of the network can only be found by considering the network as a whole.

In this section we will discuss an alternative energy function, for which we can show that, given a certain choice of thresholds and weights, the state of the network that represents the correct parse of the input sentence gives the unique, global minimum energy. The alternative energy function is given by

$$E' = \sum_k E'_{loc,k} \qquad (5.3)$$

in which,

$$E'_{loc,k} = \begin{cases} +1 & \text{iff } E_{loc,k} \geq 0 \\ -1 & \text{iff } E_{loc,k} < 0 \end{cases} \qquad (5.4)$$

where $E_{loc,k}$ is given by (5.2). From (5.3) it follows that the global minimum value of the energy of a network consisting of N units is equal to $-N$.

Let us now consider our connectionist parsing system. We introduce the following terminology:

- A connectionist primitive is *happy* iff all its units are on (output is +1), or all its units are off (output −1),
- A pair of binder units is *happy* iff at most one of them is on.

Using this terminology, it follows that a state of the network that represents a correct parse consists of only happy primitives and happy pairs of binder units; and in all oth-

er states some of them will be unhappy. As we will see below, it is possible to set the thresholds and the weights in the network such that both in a happy primitive as in a happy binder pair, all units contribute -1 to the energy, and if a primitive or a binder pair is unhappy then at least one of its units will contribute $+1$ to the total energy. So the state that represents the correct parse will be the only state with a global minimum energy of $-N$. (We will assume that the context-free grammar on which the network is based is unambiguous; therefore there will be only one correct parse tree for each input sentence.)

We will now discuss an example to demonstrate that weights and thresholds can be chosen such that all units of a connectionist primitive or of a binder pair will contribute -1 to the energy if and only if the primitive or the binder pair is happy. Therefore, we consider the network given in figure 5.1a. This network contains two types of primitives and a binder pair in a typical configuration. In figure 5.1b we show a state of the network in which the primitives and the binder pair is happy; from (5.3) it follows that the energy is -7. Figure 5.1c gives a state of the network with two unhappy primitives; its energy is -1. Considering all other possible states shows that only when both the primitives and the binder pair are happy the energy, E', is -7.[1]

In this section we discussed how, using an alternative energy function, one can design a parsing network with a unique energy minimum that corresponds to a state of the network that represents the correct parse of the input sentence. The design only requires a local analysis of small parts of the network. Because, the $E_{loc,k}$ in (5.2) is not a binary-valued function, a similar approach does not work for the energy function given in (5.1).

We can use the alternative energy function in a simulated annealing scheme to find the global energy minimum, because such a scheme is independent from the specific form of the energy function (Kirkpatrick et al. 1983). In the simulated annealing scheme we use the updating rule given in (2.7). To apply this rule we have to calculate ΔE_j; the energy difference between a state with $s_j = -1$ and that state with $s_j = +1$. From (5.3) it follows that for the energy, E', this quantity can't be calculated on a strictly local basis, that is from the output values of the units directly linked to unit j, the weights on these links and the threshold of unit j (equation (2.5)); instead, we have to consider also the next-nearest neighbors, that is the units linked to the units that are directly connected to unit j. This is a major drawback for the applicability of the energy function given by (5.3).

---

[1] This is a slightly simplified description of the actual situation; in the state with the global minimum energy all binder pairs and primitives must be happy, but also all triples consisting of a binder pair and its 'parent' unit must be in a state such that when the parent unit is on, one of the binders must be on, and when the parent unit is off, both binders must be off.

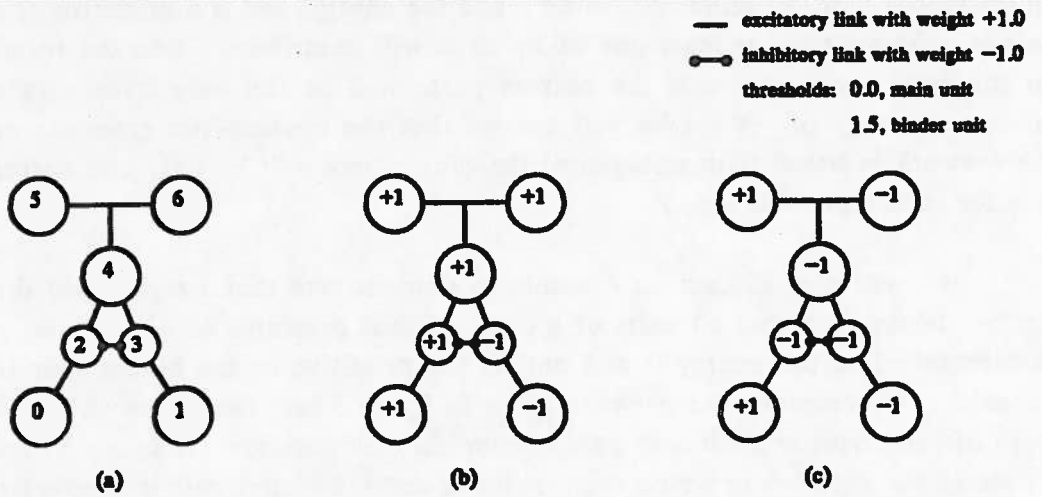


Figure 5.1 Part (a) gives the setting of weights and thresholds such that the global energy minimum is $-N$. Parts (b) and (c) give two states of the network with E' respectively -7 and -1.

5.3 The −1/+1 model

Hinton and Sejnowski (1983b) show how the weights and the thresholds in a connectionist model using the Boltzmann computational scheme are directly related to the mutual dependency of the hypotheses that are represented by the computing units, expressed in terms of conditional probabilities. For the hypotheses h and e, represented by two connected computing units, they find at $T = 1$:

$$w_{he} = \ln \frac{p(e|h)}{p(e|\bar{h})} \qquad \theta_h = -\ln \frac{p(h)}{p(\bar{h})} \qquad (5.5)$$

in which $p(h)$ is the probability that hypothesis h is *true*, which is identified with the probability that the computing unit representing h has output value +1; $p(e|h)$ is the probability that e is *true* given that h is *true*; and $\bar{h}$ is the negation of h. From (5.5) we see that the negation of e has no effect on h, and this also follows from the updating rule (2.7); a computing unit with output equal to zero does not contribute to the energy of its neighbors, and has therefore no influence on them. Hinton and Sejnowski solve this problem by changing the threshold of unit h:

$$\theta_h = -\ln \frac{p(h)}{p(\bar{h})} - w_{h\bar{e}} \qquad (5.6)$$

in which

$$w_{h\bar{e}} = \ln \frac{p(\bar{e}|h)}{p(\bar{e}|\bar{h})} \qquad (5.7)$$

When $s_e = +1$ we have to compensate for this change; this can be done by changing the weight on the link between the units:

$$w = w_{he} - w_{h\bar{e}} \qquad (5.8)$$

We propose a more direct way to represent the influence negation of e on h, namely by using -1 instead of $+1$ as an output value of the computing units. We will show that there exists a one-to-one mapping between both models; and that the use of -1 facilitates the representation of a symmetrical dependency relation (that is that the influence of e on h is the of the same as of $\bar{e}$ on $\bar{h}$).

First, we will consider the mapping between both models. Suppose we have two networks A and A' with symmetrical connections. A is given by $\{\{s_i\}, \{w_{ij}\}, \{\theta_i\}\}$ in which s_i is 0 or $+1$, and A' is defined by $\{\{s'_i\}, \{w'_{ij}\}, \{\theta'_i\}\}$ in which s'_i is -1 or $+1$. Both networks have N computing units. As we will show below, these two networks, both using the updating rule given in equation (2.7), will behave exactly the same if

$$\begin{aligned} s'_i &= 2\, s_i - 1 && i = 1, \ldots, N \\ w'_{ij} &= w_{ij} && i, j = 1, \ldots, N \\ T' &= 2\, T \\ \theta'_i &= 2\, \theta_i - \sum_j w_{ij} && i = 1, \ldots, N \end{aligned} \qquad (5.9)$$

That A and A' behave in the same way follows from the following observation. Both systems behave in such a way that they minimize their energy. Equation (2.4) gives the energy of system A, and similarly the energy for system A' is given by[1]

$$E' = -1/2 \sum_{ij} w'_{ij} s'_i s'_j + \sum_i \theta'_i s'_i. \qquad (5.10)$$

To minimize this energy, both models use the updating rule given in equation (2.7); ΔE_j plays a central role in the updating rule. In the $-1/+1$ model, this quantity, the energy difference between a state with $s_j = -1$ and a state with $s_j = +1$, in networks with symmetrical connections is given by

---

[1] One should not confuse the E' defined here as the energy function of the $-1/+1$ model, with the E' introduced in section 5.2.

$$\Delta E'_j = 2\left(\sum_i w'_{ij} s'_i - \theta'_j\right) \qquad (5.11)$$

Using (5.9) it follows that

$$\Delta E'_j = 2\,\Delta E_j \qquad (5.12)$$

Therefore at a temperature $T' = 2T$, a unit in network A' will behave in exactly the same manner as the corresponding unit in network A. (Because the temperature is just a formal parameter, the difference in T' and T is irrelevant.) So starting at the same initial configuration, that is

$$s'_i = 2 s_i - 1 \qquad i = 1, \ldots, N \qquad (5.13)$$

after a equal number of updates in network A at T_1 and network A' at $T = 2T_1$ equation (5.13) will still hold.

The advantage of the $-1/+1$ scheme follows when we consider a symmetrical dependency relation between two hypotheses h and e of the following form:

$$p(h|e) = p(\bar{h}|\bar{e}) \qquad (5.14)$$

We will assume that $p(h) = p(\bar{h})$ (that is, no a priori knowledge whether h is true or false). This assumption will simplify our calculation, but is not essential for our argument. In the $0/+1$ model it follows from equation (5.6) that

$$\theta_h = -w_{h\bar{e}} \qquad (5.15)$$

Using equations (5.9) and (5.8) we find for the $-1/+1$ model:

$$\begin{aligned} \theta'_h &= -w_{h\bar{e}} - w_{he} \\ &= -\ln \frac{p(e|h)\,p(\bar{e}|h)}{p(e|\bar{h})\,p(\bar{e}|\bar{h})} \end{aligned} \qquad (5.16)$$

Now, it follows from condition (5.14) and, because $p(\bar{e}h|h)$ is equal to $1-p(e|h)$ and $p(e|\bar{h})$ is equal to $1-p(\bar{e}|\bar{h})$, that $\theta'_h = 0$.

So, as one would intuitively expect, the threshold of unit h can be set to zero in the $-1/+1$ model when the dependency relation between e and h is symmetrical. This property was particularly useful in our parsing scheme, because for the connectionist primitives equation (5.14) holds and also $p(h) = p(\bar{h})$. Therefore the thresholds of most of the main units can be set to zero.

CHAPTER 6

Discussion

6.1 Conclusions

The major limitation of previously proposed connectionist schemes for NLU is their limited capability to do rule-based processing. We proposed a scheme that incorporates rule-based processing in a general manner. This scheme uses pairs of intermediate units to express the possible bindings between the units that represent the concepts in the connectionist model. One can choose the thresholds of the pairs of intermediate units and the weights on the inhibitory links in between them such that they function as three-state switches (both units on being a 'forbidden' state). These pairs linked together in a binary tree structure can be used to select a particular rule (for example, a grammar rule) out of a collection of alternatives. During the search for an optimal match between input data and the internal constraints in the network, the binder pairs select different rules to test whether they should be used. Interestingly, this bears a close resemblance to how a sequential processing scheme tries rule after rule; the advantage of the connectionist scheme is that many rules, each part of a different collection and represented in different parts of the network, can be applied in parallel, and also there is a complete integration of bottom-up and top-down processing.

To illustrate the applicability of our model, we discussed an example network for parsing based on a set of context-free grammar rules, a traditionally sequential form of processing. We saw that the special properties of the computational scheme of the Boltzmann machine made it possible to set the weights and thresholds by analyzing the energy of small groups of units and some general symmetry considerations. Another useful aspect of the Boltzmann scheme is that the network at temperatures just above the freezing point visits a number of states with energies equal or close to the global energy minimum of the network. Such states will, in general, show only minor differences from the state of the network that represents the correct parse; this makes the network less dependent on the particular choice of weights and thresholds.

We also discussed two modifications of the Boltzmann machine. One modification is to use -1 instead of 0 as one of the output values of a computing unit. This facilitates the implementation of symmetrical dependency relations between concepts, which enables us to set the thresholds of most units representing syntactic categories in our example parsing network to zero. Another modification we considered was the use of an alternative energy function. Using this function, it becomes relatively straightforward to choose weights and thresholds in a parsing network so that the global minimum energy of the network is unique, and corresponds to the correct parse of the input sentence. A disadvantage of this energy function is that one has to consider the output values of next-nearest neighbors of a computing unit in order to calculate the change in the energy of the network, when the output of that computing unit changes its value. Therefore, a computational scheme of Boltzmann machine using this alternative energy function will be much slower than a scheme using the original energy function.

The next logical step in this research is the addition of a semantic component to our scheme, to extend the disambiguation capability. Such a model would incorporate rules for case filling.

6.2 Some open questions

In this section we will discuss some interesting open questions in the use of connectionism in AI.[1] The emphasis will be on the use of the Boltzmann scheme.

- **Search.** One of the most important aspects of Boltzmann-like architectures is the fact that they provide us with a new, potentially very efficient, constraint-satisfaction search technique. The efficiency is closely related to the particular shape of the energy surface of the network. This shape is determined by the set of weights and thresholds representing the knowledge stored in the system. Thus, there is a direct relation between the setting of weights and thresholds and the efficiency of the computational scheme. In general, the setting of weights and thresholds is a strongly underdetermined problem; that is, many possible settings will represent the set of internal constraints that must be stored in the network. (For example, in our parsing network the weights were set according to rules containing constants that could be chosen freely.) One could therefore tune the set of weights and thresholds in order to increase the efficiency of the search. Note that, when using a distributed representation, one also has a considerable freedom in choosing the sets of units that represent

[1] Issues concerning the application of connectionism in NLU have been discussed in chapter 3.

the different concepts in the scheme (Hinton and Sejnowski 1984).[2]

• **Knowledge representation and reasoning.** It is generally accepted that efficient knowledge representation schemes should incorporate some reasoning capabilities. However, there are definite theoretical limits to what formal reasoning techniques can accomplish. Unfortunately, the requirements of sophisticated knowledge-based systems go well beyond these limits (Levesque 1984). It would be very interesting to explore the applicability of a massively parallel non-deterministic search algorithm, such as the Monte-Carlo method as is used in the Boltzmann Machine, to resolve this problem. Such an algorithm would not guarantee an solution within reasonable time limits, but would have a high probability of finding a good approximate solution in a fixed period of time.

• **Sequential processing.** Hinton, Sejnowski, and Ackley (1984) point out that there is a fundamental problem to model sequential symbolic processing with the Boltzmann machine; the optimal match between input data and internal constraints is given by the system at thermal equilibrium, when no consistent sequences exist. They propose a system, composed of a number of internally symmetrical modules that are asymmetrically connected to one another, to model sequential symbolic processing (for example production systems). Such a system would combine parallel and sequential processing. This raises the question of to what extent processing that seems to require sets of rules should be modeled in a sequential manner. Based on the positive experience with our parsing system, we believe that in addition to parsing other forms of traditionally sequential rule-based processing can be dealt with in a completely parallel manner.

• **Learning.** An important issue with respect to the learning algorithm for the Boltzmann machine is its speed. Experiments have shown that the algorithm is quite slow, even when tested on small-scale examples (Hinton, Sejnowski, and Ackley, 1984). In these experiments one starts off with completely 'blank' networks, and many iterations are needed to determine regularities in the training set. An interesting alternative to explore is to start off with networks containing some 'raw structure' and to consequently tune the network in such a way that it efficiently represents a certain body of knowledge (Hinton et al. briefly discuss the possibilities of 'one-shot' learning of facts). The raw structure might be obtained by considering the energy of small groups of units in the network, like we did in our parsing scheme.

---

[1] In the localist approach each concept is represented by one unit; however, as we saw in section 4.3, in our parsing scheme one still has some freedom in the design of the network.

References

BALLARD, D.H. and HAYES, P.J. "Parallel Logical Inference." *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*, Boulder, Colorado, June 1984, 286-292.

BINDER, K. (Ed.) (1978) *The Monte-Carlo Method in Statistical Physics*. New York: Springer-Verlag, 1978.

CHARNIAK, E. (1983) "Passing markers: A Theory of contextual influence in language comprehension." *Cognitive Science*, 7(3), July-September 1983, 171-190

CHARNIAK, E., GAVIN, M. K., and HENDLER, J.A. (1983). "The Frail/NASL reference manual." Technical report CS-83-06, Department of Computer Science, Brown University, Providence, RI 02912, February 1983.

COLLINS, A.M., and LOFTUS, E.F. (1975). "A spreading-activation theory of semantic processing." *Psychological Review*, November 1975, 82, 407-429.

COLLINS, A.M., and QUILLIAN, M.R. (1972). "Experiments on semantic memory and language comprehension." in L.W.Gregg (Ed.), *Cognition in learning and memory*. New York: Wiley, 1972.

COTTRELL, G.W. and SMALL, S.L. (1983). "A connectionist scheme for modeling word sense disambiguation." *Cognition and Brain Theory*, 6(1), 1983, 89-120.

DEESE, J. (1961). "From the isolated verbal unit to connected discourse." in: C.N. Cofer (Ed.), *Verbal Learning and Verbal Behavior*. New York: McGraw-Hill, 1961, 11-41.

FAHLMAN, S.E. (1979). *NETL: A system for representing and using real-world knowledge*. Cambridge, Massachusetts: The MIT press, 1979.

FAHLMAN, S.E. (1981). "Representing Implicit Knowledge." in: G.E.Hinton and J.A.Anderson (Eds.) *Parallel Models of Associative Memory*. Hillsdale, NJ: Erlbaum, 1981.

FAHLMAN, S.E. (1982). "Three flavors of parallelism." in *Proceedings of the Fourth National Conference of the Canadian Society for the Computational Studies of Intelligence*. Saskatoon, Saskatchewan, May 1982.

FAHLMAN, S.E.; HINTON, G.E. and SEJNOWSKI, T.J. (1983). "Massively parallel architectures for AI: NETL, Thistle, and Boltzmann machines.", *Proceedings of the National Conference on Artificial Intelligence (AAAI-83)*, Washington, August 1983, 109-113.

FELDMAN, J.A. (1982). "Dynamic Connections in Neural Networks." *Biological Cybernetics*, **46**, 1982.

FELDMAN, J.A. and BALLARD, D.H. (1982). "Connectionist Models and Their Properties." *Cognitive Science*, **6**, 1982, 205-254.

GENTNER, D. (1982). "Some interesting differences between nouns and verbs." *Cognition and Brain Theory*, **4** (2) 1982, 155-184.

HINTON, G.E. (1981). "Implementing semantic networks in parallel hardware." in: G.E.Hinton and J.A.Anderson (Eds.) *Parallel Models of Associative Memory*. Hillsdale, NJ: Erlbaum, 1981.

HINTON, G.E. (1984). "Distributed Representations." Technical report CMU-CS-84-157, Pittsburgh, PA: Carnegie-Mellon University, October 1984.

HINTON, G.E. and SEJNOWSKI, T.J. (1983a). "Analyzing cooperative computation." *Proceedings of the Fifth Annual Conference of the Cognitive Science Society*, Rochester, NY, May 1983.

HINTON, G.E. and SEJNOWSKI, T.J. (1983b). "Optimal perceptual inference." *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington DC, June 1983.

HINTON, G.E. and SEJNOWSKI, T.J. (1984). "Learning semantic features." *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*, Boulder, Colorado, June 1984, 286-292.

HINTON, G.E., SEJNOWSKI, T.J. and ACKLEY D.H. (1984). "Boltzmann Machines: Constraint Satisfaction Networks that Learn." Technical report CMU-CS-84-119, Pittsburgh, PA: Carnegie-Mellon University, May 1984.

HIRST, G. (1983). "Semantic interpretation against ambiguity." Ph.D. thesis, Department of Computer Science of the Brown University, December 1983. To appear in the series *Studies in Natural Language Processing*, Cambridge University Press, 1985.

HOPFIELD, J.J. (1982). "Neural networks and physical systems with emergent collective computational abilities." *Proceedings of the National Academy of Sciences USA*,**79**, 1982, 2554-2558.

KIRKPATRICK, S.; GELATT, C.D.Jr. and VECCHI, M.P. (1983). "Optimization by simulated annealing." *Science*, **220**#4598, 1983, 671-680.

LEVESQUE, H.J. (1984), "A Fundamental Tradeoff in Knowledge Representation and Reasoning." *Proceedings of the Fifth Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, London, Ontario, Canada, May 1984.

LESSER, V.R. and CORKILL, D.D. (1983). "The Distributed Vehicle Monitoring Testbed." *The AI Magazine*, 4 (#3), 1983, 15-34.

McCLELLAND, J.L. and RUMELHART, D.E. (1981). "An interactive activation model of context effects in letter perception. Part 1, An account of basic findings." *Psychological Review*, 88, 1981, 375-407.

METROPOLIS, N., ROSENBLUTH, A.W., ROSENBLUTH, M.N., TELLER, A.H., and TELLER, E. (1953). "Equation of State Calculation by Fast Computing Machines." *Journal of Chemical Physics*, 1953, 21, 1087.

MINSKY, M., and PAPERT, S. *Perceptrons*. Cambridge, MA: MIT Press, 1968.

PALMER, S.E. (1975). "Visual perception and world knowledge: Notes on a model of sensory-cognitive interaction." in: D.A.Norman and D.E. Rumelhart (Eds.) *Exploration in Cognition*. San Francisco: Freeman, 1975.

POLLACK, J.B. and WALTZ, D.L. (1982). "Natural language processing using spreading activation and lateral inhibition." *Proceedings of the fourth Annual Conference of the Cognitive Science Society*, Ann Arbor, August 1982, 50-53.

POSNER, M.I. (1978). *Chronometric Explorations of Mind*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1978.

QUILLIAN, M.R. (1968). "Semantic memory." in: MINSKY, M.L. (ed.). *Semantic Information Processing*. Cambridge, Massachusetts: The MIT Press, 1968, 227- 270.

QUILLIAN, M.R. (1969). "The teachable language comprehender: A simulation program and theory of language." *Communications of the ACM*, 12(8), August 1969, 459-476.

REILLY, R.G. (1984). "A connectionist model of some aspects of anaphor resolution." *Proceedings of the 10th International Conference on Computational Linguistics*, Stanford, July 1984, 144-149.

RILEY, M.S. and SMOLENSKY, P. (1984) "A parallel model of (sequential) problem solving." *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*, Boulder, Colorado, June 1984, 286-292.

SCHANK, R.C. (1975). *Conceptual Information Processing*. Amsterdam: North-Holland publishing company, 1975.

SMALL, S.L.; COTTRELL, G. and SHASTRI, L. (1982). "Towards Connectionist Parsing." *Proceedings of the National Conference on Artificial Intelligence (AAAI-82)*, Pittsburgh, August 1982, 247-250.

SMALL, S.L., SHASTRI, L. BRUCKS, M.L., KAUFMAN, S.G., COTTRELL, G.W., and ADDANKI, S. (1983). "ISCON: a Network Construction Aid and Simulator for Connectionist Models." Technical report 109, Computer Science Department, University of Rochester, April 1983.

STABLER, E.P., Jr. (1983). "How are grammars represented?" *The Behavioral and Brain sciences*, 6, 1983, 391-421.

VON NEUMANN, J. (1958). *The Computer and The Brain*. New Haven: Yale University Press, 1958.

WALTZ, D.L. and POLLACK, J.B. (1984). "Phenomenologically plausible parsing." *Proceedings of the National Conference on Artificial Intelligence (AAAI-84)*, Austin, Texas, U.S.A., August 1984, 335-339.

WINOGRAD, T. (1983). *Language as a cognitive process*. Reading, MA: Addison-Wesley Publishing Company, 1983.

WINSTON, P.H. (1984). *Artificial Intelligence*. (2nd edition) Reading, MA: Addison-Wesley Publishing Company, 1984.