INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

# Towards Computational Models of the Brain: Getting Started

Jiří Wiedermann

Technical report No. V–678

June 1996

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic
phone: (+422) 6605 3520   fax: (+422) 8585789
e-mail: wieder@uivt.cas.cz

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

# Towards Computational Models of the Brain: Getting Started

Jiří Wiedermann

Technical report No. V–678
June 1996

## Abstract

[1]We present an overview of known results from machine oriented complexity theory which relates to brain–like computing. First, a brief survey of complexity results on various kinds of finite neural networks is given. The lower estimate on their computational and descriptive power is obtained with the help of so–called *neuromata.* Then, subsequently, three computational models of brain are presented: Valiant's *neuroidal tabula rasa*, the *memory surface model* by Goldschlager, and de Bruijn's model of a *brain as a molecular computer.* These three models are compared and their weakness as well as advantages are discussed. Finally, the prospects of computer simulation of the above models of brain are

**Keywords**
Computational Models; Neurocomputing; Brain–Like Computing; Molecular Computing

[1]This research was supported by GA ČR Grant No. 201/95/0976 "HYPERCOMPLEX"

# Contents

## 1 Introduction

Thinking about the brain as being a computational device that can be mathematically formalized and explained presents an exciting problem which has been with us since the dawn of computer science. In the following, under the notion of brain–like computing, we shall understand any computing whose main principles are inspired by our ideas on how brains operate. Thus, we shall view the brain as a computing device, or a transducer, to be precise, that requires input data, and produces output data, both encoded in a suitable way. This is in contrast with a robotic approach where sensors and effectors are considered as parts of the system.

To speak about brain–like computing is definitely a less ambitious project than to speak about computational models of the brain. Namely, in the former case, we basically aim only at modelling, investigating and explaining some basic or partial computational phenomena that brains are capable of, while in the latter case, explanation (or theory) of complex brain behaviour in computational terms is to be expected.

In understanding the brain, nature has confronted us with different roadblocks, which in their variety and entirety, have not been challenged successfully either in computer, or in any other science.

The nature of the above mentioned barriers is diverse. To overcome them one would ideally like to at least see a kind of three level model of a brain similar to the case of any abstract computational system (cf. [25]).

At the highest "machine independent" level one needs some kind of functional specifications stating at a fairly high level, but as formally as possible, what information–processing task is to be modeled. Informally this is described as "higher brain functions" in other disciplines like psychology.

Then at the next lower level one would need a more refined model dealing with only several basic brain operations out of which more complex mental operations on the previous level can be built. It is perhaps here where the computer science can help in identifying this elementary set of brain operations. Of course, other sciences, like psychology and neurobiology can offer a valuable guide in this search.

Finally, one needs a plausible "machine" model of a brain in which the basic operations can be efficiently realized. Here again neurobiology and neuroanatomy, together with machine oriented complexity theory, can be of tremendous help.

No satisfactory specifications of any of the above mentioned three levels are known. Presently it seems that the key to understanding the brain and its efficient implementation is hidden exactly in the second, middle level that presents a kind of bridge between higher level brain functions as described in the first level, and the machine model from the third level.

Roughly speaking, brain–like computational models that we shall deal with in the sequel will each concentrate on a different level or on different levels with a various intensity in details, in the above mentioned sketch of a three level brain architecture. In all cases there will be at least some loose reference to the remaining levels. None of these models seem to be able to satisfactorily cover all three levels to a full extent. Nevertheless, each of them contains perhaps some valuable idea(s) that can help in designing better models. Of course, in computational models of brain we are not striving for a biological faithfulness — rather we are after the operational faithfulness, at least to some interesting level. In doing so, biological principles can serve as a a welcomed inspiration.

The organization of the survey is as follows.

In the second section, by concentrating on main "sensor independent" aspects of the brain, we shall introduce a simplified basic view of the brain as a transducer.

In the third section we shall describe the first and simplest model of a brain, viz. that of finite neural nets. This model is composed of a finite number of individual discrete neurons of so–called first generation. As presented, this model does not intend to model any other interesting cognitive phenomena except of a fast recognition. Nevertheless it is simple enough to enable the investigation of the descriptional and computational power and efficiency of the respective networks with the help of so–called *neuromata* [20]. In fact, most of the other models can be seen as more elaborated versions of this basic model. We shall also pay some attention to the second and third generation of neural nets (those with analog, and spiking neurons, respectively).

The next model we shall describe in the fourth section is that of *neuroidal tabula rasa* by Valiant [22]. It already presents quite an elaborate model of the brain that is able to explain e.g. the memory allocation problem, and suggests the implementation of basic cognitive tasks like concept learning, their combination into more complex knowledge representations, and some other tasks. Thus, this model describes the lowest and the middle level of the above mentioned three level architecture.

Goldschlager's model of *memory surface* from [7] described in Section 5 is the model that spans over all three levels. It is able to provide a plausible explanation not only for some basic cognitive tasks, like formation of abstract concepts, association of ideas and train of thoughts, but also for some higher brain functions, like creativity, self, consciousness and free will. Even a theory of sleep is presented which is consistent with this model. Unfortunately, this model seems to be less explicit than the previous models w.r.t. description of the lowest, machine model.

The last model described here is that of *thinking soup and roaming random set* by de Bruijn [3] from Section 6. This is an exceptional model that concentrates mainly on certain aspects of brain computing — mainly on associative storage and retrieval, and consciousness. In its basic assumptions about the principles of brain operating it differs substantially from all previously mentioned models, since it makes use of a molecular computing. However, the author gives no concrete references from biology, let us say, that would support these principles. This approach seems to be a good example of an alternative, speculative approach to brain computations based on not yet quite exposed, but to some extent plausible, assumptions.

In Section 7 we shall compare the previous models with each other in order to uncover their bottlenecks and superiorities. Also, the possibilities of implementing brain computations according to three different scenarios will be discussed.

As a suitable expository reading of results related to brains, machines, and mathematics up to nineteen eighties the book [1] by Arbib is to be recommended; for the recent overview, see [2] by the same author. The biological aspects in brain–like computing are studied in the book by Churchland–Sejnowski [4].

## 2 Brain as a Transducer

In what follows, we shall be essentially interested in the brain–like computing that is performed, or inspired, by the actions that happen only in a particular area of brains — namely in those areas that already obtain some encoded, simplified, selected inputs and that produce specific signals that should still be interpreted by other parts of brains.

Thus, in order to understand what the most fundamental tasks, which a brain can handle, are and how these are implemented, it is helpful to view the brain as an *input/output transducer*, whose purpose is to interact with the environment. Signals arriving from the receptors are first *preprocessed* by special purpose brain components and arrive into a special "central" part of the brain where they are processed in a manner that will be the subject of our further concern. Signals, leaving this special part, enter into *postprocessors* that connect directly to *effectors* which carry out the commands issued by postprocessors.

This view of the brain as a transducer, omitting details of pre– and post–processing, is common to all models that we shall deal with in the sequel. Depending on the model, the above mentioned "central" part of the brain is modeled by neural nets, neuroidal tabula rasa, memory surface, etc.

## 3 Neural nets

### 3.1 Motivation

Where does one begin with the study of brain–like computing? It is here, where the neurobiology provides us with a useful hint: let us begin with an investigation of so–called neural nets. The basic computational element of such nets is presented by the model of an *artificial neuron*, that was designed by McCulloch and Pitts in 1943 [12], as an idealized model of real biological neurons occurring in brains. Despite its simplicity, the authors have shown that the finite control of Turing machines (that have been known for less than a decade) can be realized with the help of neural nets. This was the first evidence of the computational potential of neural nets. Turing, himself, was aware of this result. In 1945 Turing and von Neumann, respectively, used the neurons as the building blocks for the construction of universal digital computers [6].

From a historical point of view, it is interesting to note that the invention of a formal neuron has preceded that of a finite automaton, which was only defined in mid fifties by Kleene [10]. Essentially, he defined it by mathematically reworking the ideas of McCulloch and Pitts [14].

McCulloch and Pitts's model of a neuron that only allows Boolean inputs and outputs, is nowadays called *the first generation neuron*. The computational characteristics of the respective neural nets will be described in the sequel (part 3.2). Nowadays, there are also other kinds of neurons studied, which will be briefly mentioned in subsection 3.3.

## 3.2 First Generation Neural Nets

In the following we shall formally define neural nets in a way that, as we shall later see in Section 4, will be compatible with the definition of neuroidal nets. We shall first formally define the structure of neural nets, and then their computational behaviour.

**Definition 3.1** A (first generation) neural net $\mathcal{N}$ *a quartuple* $\mathcal{N} = (\mathcal{G}, \mathcal{W}, \mathcal{T}, \delta)$, *where*

- $G = (V, E)$ *is the directed graph describing the topology of the network; $V$ is a finite set of $N$ nodes called* neurons *labeled by distinct integers* $1, 2, \ldots, N$, *and $E$ is a set of directed edges between the nodes. The edge $(i, j)$ for $i, j \in \{1, \ldots, N\}$ is an edge directed from node $i$ to node $j$.*

- $W$ *is the finite set of integers that are called* weights. *Each edge $(i, j) \in E$ has a value $w_{i,j} \in W$.*

- $T$ *is the finite set of integers called* thresholds. *Each neuron $i$ has assigned a threshold $t_i \in T$.*

- $\delta$ *is the* state update function *of form* $\delta : Q \times S \to Q$, *where*

  - $Q = \{0, 1\}$ *is the set of two states called* firing *and* quiescent *state, respectively;*
  - $S$ *is the finite set of all possible sums of weights of all neurons. I.e., the members of $S$ are obtained by taking the sums of all the possible $N$–tuples (with repetitions) with component values taken from $W$.*

  *Let $w_i$ be the sum of those weights $w_{ki}$ of neuron $i$, which are on edges $(k, i)$ that come from neurons that are currently in firing states, i.e., formally*

$$w_i = \sum_{\substack{k \text{ firing} \\ (k,i) \in E}} w_{ki}$$

  *The state update function $\delta$ is defined as follows:*

$$\delta(0, w_i) = 1 \text{ iff } w_i \geq t_i$$
$$\delta(0, w_i) = 0 \text{ iff } w_i < t_i$$
$$\delta(1, w_i) = 1 \text{ iff } w_i \geq t_i$$
$$\delta(1, w_i) = 0 \text{ iff } w_i < t_i$$

The *computation of a neural network* is determined by the *initial conditions* and by the *input sequence*. The initial conditions specify the initial values of states of the neurons in the network at the beginning of computation.

The input sequence is a sequence of neuron sets specifying, for each $t = 0, 1, 2, \ldots$, the neuron sets that are forced to fire (or prevented from firing) at that moment by mechanisms outside of the net (by peripherals, say). The respective set is called the *set of input neurons* at time $t$.

Analogously, there is *the output sequence* which is defined as a sequence of neuron sets specifying, for each $t = 0, 1, 2, \ldots$, the neuron sets the firing of which will represent an output of the computation at the respective time. The respective set is called the *set of output neurons* at time $t$.

The network then starts to compute synchronously, by updating the states of all neurons in the network simultaneously in accordance with the definition of the respective state update functions $\delta$. At each time the new subset of neurons from the input sequence is set to the prescribed values (a new instance of input is read). The result of computation at time $t = 0, 1, 2, \ldots$ is determined by firings of output neurons corresponding to time $t$.

## Neural Automata

The previous definition of a neural net is fairly general in the sense that it allows multiple inputs and outputs, that, moreover, can take place in various neurons in accordance with the input and output sequence.

In order to intuit the computational potential and limits of first generation neural nets we shall concentrate on the most simplest case of input/output schedule of such nets. In this case we shall have exactly one fixed input neuron $u$ and exactly one fixed output neuron $v$, with $u \neq v$. The corresponding neural net is then called *a neural automaton*, or shortly *a neuromaton*.

It can be seen as a language recognizer that recognizes words over the alphabet $\{0, 1\}$. The input sequence is read via the input neuron, bit by bit, and with a possible delay the output neuron signals whether the input sequence, read so far, is accepted.

These devices have been studied in detail in [19], [20] and [26] where most of the results we shall be referring to originate.

First of all, it appears that one can restrict the attention to neuromata reporting the acceptance/nonacceptance with a unit delay. This is a rather non–trivial result, since it requires showing that to each neuromaton with a greater delay there exists an equivalent — albeit larger — neuromaton with a unit delay. The new neuromaton is larger by a multiplicative factor that is exponential w.r.t the original delay.

It is not surprising that neuromata precisely recognize the class of regular languages — this was basically already shown in [10]. Moreover, any regular language given by a regular expression of length $n$ is recognized by a neuromaton of size $\Theta(n)$ (i.e., consisting of that number of neurons) and in general this size cannot be decreased. The descriptional complexity (i.e., the total number of bits required to completely specify the neuromaton) is quadratic. On the other hand, there are examples of neuromata of size $\Theta(n)$ and of descriptional complexity $\Theta(n^2)$ (because of their exponential weights) that recognize regular languages described by regular expression that cannot be shorter than $\Omega(2^n)$ (a possible example is a neuromaton recognizing all words of length $2^n$). This seems to point to the greater expressive power of neuromata when compared to regular expressions. This result is also confirmed by the interesting fact that the emptiness problem for a (regular) language given by the corresponding neuromaton is PSPACE–complete. This is to be compared with the similar problem for regular expression or finite automata, which is known to be only NLOGSPACE–complete.

Also various restricted versions of neuromata have been studied.

First, it has been shown that neuromata can be used for pattern matching. For a pattern of length $n$ there exists a "neuro–pattern–matcher" of size $\Theta(\sqrt{n})$ and either with $\Theta(\sqrt{n})$ connections with weights of size $\Theta(2^{\sqrt{n}})$, or with $\Theta(n)$ connections of constant size weights. Thus, in both cases the construction is optimal as far as its descriptional complexity is concerned.

Second, it follows from other known results that neuromata of a respective size $\Theta(2^{n/2})$ and $O(\sqrt{n})$, each with constant size weights, exist that respectively compute the values of an arbitrary, or symmetric, Boolean function of $n$ arguments (cf. [13]).

Third, a special class of so–called Hopfield neuromata has been defined and studied. In these neuromata the connections among neurons are symmetric and each symmetric pair carries the same weight. It follows that the underlying neural network is a Hopfield network. It has been shown that Hopfield neuromata recognize a proper subclass of regular languages. A complete characterization of the respective Hopfield languages in syntactic terms, concerning the corresponding regular expressions, has been discovered [18]. It is a non–trivial result that builds on the known convergence properties of Hopfield neural networks (cf. [13] or [23]).

In the framework of brain–like computing, the previous results can be interpreted as the ones expressing that neuromata present a device that can realize the basic cognitive task of recognition, with a constant delay and with complexity characteristics, depending on the size and number of recognized patterns. The patterns are represented as words of a (regular) language to be accepted. Boolean functions with the bounded number of arguments can be encoded into the words of such languages.

## 3.3 Second Generation Neural Nets

Second generation neural nets differ from the first generation nets in their basic processing unit. This unit — an *analog neuron* — is able to process analog inputs and to produce analog outputs. I.e., the respective variables take on continuous real values, and hence the neuron computes the values of some real function. This is achieved by applying a so–called *activation function* $\psi : \mathbb{R} \to \mathbb{R}$ to the weighted sum of inputs to the neuron at hand. The output value is, then, also a real number. Thus, if $y_k$ is the input to $i$ from a neuron that is connected to $i$ via the edge $(k, i) \in E$, carrying the weight $w_{ki}$, then the output $y_i$ of $i$ is defined as

$$y_i = \psi\Big(\sum_{(k,i)\in E} w_{ki} y_k\Big)$$

Depending on the activation function (that can be e.g. a sigmoidal function, or linear saturated function, or piecewise polynomial function) and on the weights (that can be either integers, or rationals, or reals), the computational power of the respective nets ranges from that of the first generation (regular languages) up to super–Turing capabilities. For an overview see e.g. [15], [16], or [17]. Thus, among the second generation neural nets there exist the most powerful computational devices know to us. To illustrate the computational efficiency of the respective networks, it can be shown that a finite number of analog neurons can simulate Turing machine on inputs of arbitrary length (cf. [9] or [17]). The corresponding computation relies heavily on the ability of the underlying network to compute with real numbers with arbitrary precision.

From a biological point of view networks computing with reals are unrealistic — no biological system seems to be able to cope with analog values to an arbitrary precision. This excludes the possibility that biological systems could possess the super–Turing computational capabilities.

However, the possibility that biological neurons are not completely of a discrete nature, as was the case with the first generation neurons, seems to still be still. This is supported also by the fact that some biological neurons are known to fire at various intermediate frequencies between their minimum and maximum frequency. Thus, the quest for the "right" model of a neuron, whose computational power would lay somewhere in–between the extreme of discrete computing, and the other extreme of exact computing with reals, is still being continued.

## 3.4 Third Generation Neural Nets

These nets are based on the model of *spiking neurons*. They still provide a simplified model of a real biological neuron by concentrating on just a few aspects that have not, however, been captured by the previous generation of neurons. In particular, they describe the actual output of a biological neuron much better, and hence they allow us to investigate the possibilities of using time (or timing) as a resource for computation and communication on a theoretical level. Whereas the timing was trivialized in both preceding generations (by assuming synchronousness in most of the cases), the timing of individual computational steps plays a key role for computations in networks of spiking neurons. In fact, the output of a spiking neuron $v$ consists of the set (cf. [11])

$$F_v = \{t_1, t_2, \dots, t_k | k \geq 1, v \text{ fires in time } t_i, 1 \leq i \leq k\} \subseteq \mathbb{R}^+ = \{x \in \mathbb{R} : x \geq 0\}$$

For details see e.g. the paper by Maass [11] where the computational power of networks of spiking neurons is investigated and compared with that of preceding generations. It is shown that these nets are computationally more powerful than the other neural net models. A concrete, biologically relevant function is exhibited which can be computed by a single spiking neuron, but which requires hundreds of hidden units on a sigmoidal neural net of the second generation. The paper contains an extensive list of references to the currently available literature on computations in networks of spiking neurons and relevant results from neurobiology.

Spiking neurons are a relatively new phenomenon in brain–like computing and, therefore, their computational theory is still in the early stages of development. It is true that they have apparently been suggested as a possible framework for a computational model of the brain, as we shall see in Section 5. Nevertheless, their use there was highly informal, without making any reference to a more concrete model. Therefore, we shall not dwell on the related issues any longer.

# 4 Neural Tabula Rasa

## 4.1 Introduction

First generation neural nets described and studied in the subsection 3.2 present, in the best case, a very simple model of brain that models only one fundamental cognitive task — viz. recognition. The main feature that seems to be missing is that of learning ability. At first sight it is not clear whether these nets are powerful enough to model also learning. The main obstacle, in this sense, seems to be the fact that these nets, once defined, cannot modify their computational behaviour. This seems to be a condition *sine qua non* for any learning device. This defect is eliminated in so–called *neuroidal nets* that present a programmable kind of neural nets, which was first proposed by Valiant in 1988 [21]. The potential to model brain–like computation was further investigated in 1994 on the monograph [22], by the same author. The main building element of such networks is *a neuroid*, which is a mix of a neural threshold element with the idea of finite automaton and additional features that enable a neuroid to change its state, weight, and/or threshold depending on the activities of neighbouring connecting neuroids.

## 4.2 Neuroidal Nets

Next we shall formally give the definition of neuroidal nets, in the same spirit as in the definition of neural nets. In fact, from this definition it will be apparent that neural networks are but a severely restricted kind of neuroidal nets. In the following, we shall define neuroidal nets of a somewhat specific type as compared to the original definition by Valiant [22]. Essentially, we require that the functions, which describe how the parameters of a neuron should be changed in particular situations, be all of a finite type, i.e., with both domains and ranges of values described by finite sets. This does not seem to be on the account of the practicability, since e.g. concrete neuroidal networks described, as examples, in [22] are of this restricted type.

**Definition 4.1** A neuroidal net $\mathcal{N}$ *a quintuple* $\mathcal{N} = (\mathcal{G}, \mathcal{W}, \mathcal{X}, \delta, \lambda)$, *where*

- $G = (V, E)$ *is the directed graph describing the topology of the network; $V$ is a finite set of $N$ nodes, called* neuroids, *labeled by distinct integers* $1, 2, \ldots, N$, *and $E$ is a set of directed edges between the nodes. The edge $(i, j)$ for $i, j \in \{1, \ldots, N\}$ is an edge directed from node $i$ to node $j$.*

- $W$ *is the finite set of integers which are called* weights. *Each edge $(i, j) \in E$ has at each instant of time a value $w_{i,j} \in W$.*

- $X$ *is the finite set of* modes *of neuroids that a neuroid can be in each instant. Each mode is specified as a pair $(q, T)$ of values where $q$ is a member of a set $Q$ of states, and $T$ is an integer called the* threshold *of the neuroid. $Q$ consists of two kinds of states called* firing *and* quiescent *states. To each node $i$ there is also a Boolean variable $f_i$ which has value of one or zero, depending on whether or not the node $i$ is in a firing state.*

- $\delta$ *is the* mode update function *of form $\delta : X \times S \rightarrow X$, where $S$ is the finite set of all possible sums of weights of all neuroids. I.e., the members of $S$ are obtained by taking the sums of components of all the possible $N$–tuples (with repetitions) with component values taken from $W$. Let $w_i$ be the sum of those weights $w_{ki}$ of neuroid $i$ which are on edges $(k, i)$ deriving from neuroids that are currently firing, i.e., formally*

$$w_i = \sum_{\substack{k \text{ firing} \\ (k,i) \in E}} w_{ki}$$

*The mode update function $\delta$ defines for each combination $(s_i, w_i)$ that holds at time $t$, the mode $s' \in X$ that neuroid $i$ will transit to at time $t + 1$:*

$$\delta(s_i, w_i) = s'$$

- $\lambda$ *is the* weight update function *that of form $\lambda : X \times S \times W \times \{0, 1\} \rightarrow W$. It defines for each weight $w_{ji}$ at time $t$ the weight $w'_{ji}$ to which it will transit at time $t+1$, where the new weight may depend on the values of each of $s_i$, $w_i$, $w_{ji}$, and $f_j$ at time $t$:*

$$\lambda(s_i, w_i, w_{ji}, f_j) = w'_{ji}$$

The *computation of a neuroidal network* is defined in much the same way as in the case of neural nets. The only difference is that now, at each step, both weights and modes of all neuroids in the network are updated in parallel, in accordance with the definition of the respective weight and mode update functions.

For more details about the model see [22].

## 4.3 Simulating Neuroidal Nets by Neural Nets

From the previous definitions of first generation neural nets and neuroidal nets, it is readily seen that the former really are severely restricted kinds of the latter. A natural question is whether, due to this restriction, the computational power of the neural nets is also lower than that of neuroidal ones. It appears that this is not the case. Namely, from the definition of neuroidal nets one can immediately infer that any such a net can only enter a finite (albeit probably very large) number of different configurations, that differ from each other in the value of at least one parameter (weight, threshold, or state). Therefore, the computational behaviour of any neuroidal net can be modeled by a finite automaton, which, in turn, can be simulated by a neural net, according to, let us say, the result [9] of Indyk.

The size of a simulating neural network will certainly be larger than that of the original neuroidal network. But it will essentially remain linear in both the number of neuroids, and edges of the neuroidal network, with the constant of proportionality basically depending on the size of 'programs' of individual neuroids, as it was shown in [24]. The "trick" of simulation is that in the neural net there are sets of neurons corresponding to each neuroid; to each configuration (i.e.,"setting" of its parameters) of a single neuroid there is a special neuron that simulates the neuroid in this particular configuration. As the neuroids change their parameters, the neural net "switches" among its neurons in such a way that the instantaneous configuration of the neuroidal net is modeled by a certain subset of neurons in the neural net.

This is a slightly surprising result showing that computational power of neuroidal nets is the same as that of neural nets, or, stated in other words, that the ability of neuroids to modify their own parameters in not essential. However, there is no doubt that neuroidal nets present a substantially better framework for studying learning issues than neural nets do. We shall return to the related problems in the forthcoming subsections.

## 4.4 Learning Issues in Neuroidal Nets

The ability of neuroids to change their parameters in accordance with their mode and weight update functions renders them to various learning tasks. To illustrate this in a simple example, assume that we wish to program a neuroid $i$ in such a way that after being confronted in time $t = 0$ with some inputs from its neighbours, it will "remember" those neighbours that have fired on this particular occasion, and in any future occasion (in time $t > 0$) $i$ will fire iff those neighbouring neuroids that were firing at $t = 0$ will fire again.

Assume that $i$ in time $t = 0$ is in a state we call A1 and has all weights $w_{ji}$ incoming from neighbouring nodes equal to 1. The transitions below show how the algorithm, that realizes the required updates, would be expressed within the model [22]:

$$\delta([A1, T_i], w_i) = [A2, w_i] \text{ for all } T_i, w_i$$

$$\lambda([A1, T_i], w_i, w_{ji}, 0) = 0 \text{ for all } T_i, w_i, w_{ji}$$

The first transition updates the mode. It says that if the state is A1, then the new state will be A2 and the new threshold will be $w_i$, which equals the number of presynaptic neighbours that were firing and since at time $t = 0$, each $w_{ji} = 1$. The same update occurs for all values of $T_i$.

The second transition updates the value of weight $w_{ji}$ to 0 for every $w_{ji}$ such that the given condition holds, namely that the state of $i$ is A1, and $f_j = 0$.

## 4.5 Knowledge Representation

Due to the very nature of neuroidal nets they provide a model of cognition in which all variables can be represented in the Boolean form. For these reasons in Valiant's approach cognitive tasks are formulated in a Boolean framework. Therefore, in the first instance, Boolean functions sometimes called *concepts* in this context, are used as a vehicle for knowledge representation. Boolean functions will be implemented on a neuroidal net with the help of circuits that evaluate the function or *predicate* at hand. However, as we shall outline later on, what we shall implement in reality on neuroidal nets will usually not be the exact given Boolean function, rather something that looks like the approximation of the corresponding circuit, in the following sense. Whenever we shall require that some neuroids have to be connected in the circuit, in the implementation this will be ensured only with a high probability. Hence, the correct results will be obtained also only with the high probability.

The next, central, question is to determine which classes of Boolean functions are most appropriate for modeling how (human) knowledge is represented in the brain. The simplest of such classes is that of *conjunctions* of variables. Out of conjunctions, a more general class of *disjunctive normal forms* (DNF) can be built. Surprisingly, it appears that these two classes are already enough to obtain interesting non–trivial models of brain–like computing.

## 4.6 Neural Tabula Rasa

When it comes to modeling brain computations, with the help of neuroidal nets, one has to design a concrete net topology and "program" the neuroids in the net in a way that would mimic some cognitive tasks over the inputs presented to the net. In this way we obtain a so–called *neural tabula rasa* — NTR.

In the NTR so-called *items* are represented, which describe any aspects of the reality that is modeled in the NTR. According to Valiant there are five features that should be preserved in any knowledge representation in the NTR:

(a) Each neuron corresponds to a semantic item.

(b) Typically there are several neurons representing each such item.

(c) Only those new items which are experienced and noticed by a special kind of mechanism, are added to the memory. This special mechanism Valiant calls an *attentional mechanism.*

(d) This representation is *hierarchical.* Some items can be seen as basic ones, that are preprogrammed or detected at some lower level in our perceptual system, which is external to NTR. Once some items have been assigned to the neuron, new items expressible in terms of items already represented, can be assigned to previously unused neurons.

(e) Only approximations of any idealized Boolean function are represented (see the comments on this in the previous subsection).

The items represented in NTR will be denoted by letters such as $x$, $y$, $z$, respectively. The set of neuroids representing each of these items will be $\tilde{x}$, $\tilde{y}$, $\tilde{z}$, respectively. The basic intention is that when the system is presented from the outside with an input corresponding to item $x$, then the neuroids in the set $\tilde{x}$ will fire.

The most basic aspect of knowledge representation in the NTR is that each item is stored in about $r$ neuroids, where the replication factor $r$ is viewed as a constant (say 50 or 100) for a given NTR. Replication will ensure robustness, as well as a slightly lower connectivity requirement of the network underlying the NTR, which that would otherwise be needed.

Let $E(\tilde{x})$, $E(\tilde{y})$ be the sets of nodes to which edges are directed from $\tilde{x}$, $\tilde{y}$, and call these the *directed neighbours.* We than define $E(\tilde{x}) \cap E(\tilde{y})$ to be the *frontier* of $\tilde{x}$, and $\tilde{y}$.

The notion of frontiers will be used in the description of so–called *vicinal* algorithms. Their most basic feature is that, whenever some communication has to be established between two items not directly connected, the algorithm establishes the necessary communication via neuroids from the frontier of the respective items. E.g., the frontier of $x$ and $y$ can be used to represent the conjunction $x \wedge y$.

The efficient implementation of vicinal algorithms will be, therefore, supported by special a class of graphs of form $G = (V, E)$ that possess the following, so–called $(r, l, m)$–frontier property: when $\tilde{x}, \tilde{y} \subseteq V$ are randomly chosen disjoint subsets of size $r$, the size of the frontier $E(\tilde{x}) \cap E(\tilde{y})$ has expectation $l$ and variance $m$. A graph ideal for executing vicinal algorithms would have $l = r$ and $m = 0$.

Valiant has proven that random graphs on $N$ nodes with expected degree $\sqrt{N/r}$ have the $(r, r - O(N^{-1/2}), r - O(N^{-1/2}))$–frontier property. Thus, these graphs have the expected size of any frontier of two sets about the same as their own size and, therefore, this frontier has some chance of storing a new item, which is to become an equal citizen with these two items.

Since we wish that the NTR will be able to learn hierarchically, we have to take care about the process of creating, to arbitrary depth, new frontiers from sets that were previously frontiers themselves. Unfortunately, it appears that the variance of this process is too large to maintain stability over a large number of iterations. Experiments suggest, however, that for $r = 50$, for example, this process is maintainable in the previous random graphs to dept 4 or 5 [22].

Besides the frontier properties, there is a further attribute that is required of the NTR and can be supplied by randomness. This further property is needed to ensure that the node chosen for representation of a new item will be, to a large measure, among those not previously chosen. Valiant calls this the *hashing* property since it corresponds to that notion in computer science. The corresponding theory is related to some extent to the problem of universal hashing; the results do confirm the conjecture that the previous class of random graphs can also accommodate this additional condition, to some reasonable extent.

Upon choosing a random graph, which has a good frontier and hashing property, as the topology of the NTR, one can design vicinal algorithms for implementing cognitive tasks. The task of allocating nodes for storing a conjunction $x \wedge y$ is called JOIN. Another important task is to establish *association* between the two sets $\tilde{x}$, $\tilde{z}$ already allocated. I.e., we wish to update the NTR so that, at later times, whenever $\tilde{x}$ fires, so will $\tilde{z}$. This update operation is called LINK. It is beyond the scope of this survey to describe the respective vicinal algorithms. They all work on similar principles, as illustrated in Subsection 4.4, making use of frontier and hashing properties. For the details and other cognitive tasks (e.g., memorization, correlational learning, relation expression learning, reasoning) see the Valiant's monograph [22].

# 5 Memory Surface

## 5.1 Introduction

Goldschlager's computational model of a brain [7] made its first appearance in 1984 — i.e., prior to Valiant's model (but is not mentioned in Valiant's monograph [22]). Yet, we decided to introduce this model after describing the fundamentals of neural tabula rasa, for three reasons. First, the NTR and the related issues will offer additional framework for explaining Goldschlager's ideas. Second, Goldschlager's model appears to be a model of higher level, with more elaborated parts concerning higher level brain functions, and less elaborated parts concerning the lowest level of a brain "machine" model. Here also we shall benefit from the knowledge of Valiant's ideas. And third, the memory surface model appears to be based on the third generation neurons (spiking neurons).

## 5.2 Memory Surface Model

The central part of the brain, mentioned in Section 2, which will be responsible for the processing of pre–processed signals from receptors, is called *memory surface*. It can be viewed as a directed graph comprised of millions of points, which will be called *columns*. Each column is connected to other columns nearby and no long distance connections are required. The connection between coincident columns are in both directions. Moreover, columns may have (directed, one way) connections from some pre–processing and/or to some post–processing components. For technical reasons we shall assume that each column is connected with an even number of other columns; the respective (pairs of incoming and outcoming) edges are indexed by numbers from 0 to $k-1$, for some $k$ even. There is a *weight* $m_d$ assigned to each outcoming edge (indexed by) $d$.

Columns present the basic functional unit of memory surface. All communication among columns will be in the form of *trains of pulses* running along the directed edges of the underlying graph. These pulses have the same shape and amplitude, but their frequency can vary with time.

The computational activity of each column is fully described by two kinds of characteristics: the first one is referred to as communication characteristics, the second one as memory characteristics.

The *communication characteristics* of a single column are as follows:

- whenever a pulse arrives at the column along the edge $d$, $0 \leq d < k$, a new pulse will be produced and sent out of the column along the edge $(d+k/2)\text{mod } k$ (i.e., each incoming pulse behaves as if it is travelling in a straight line through the column, providing that the edges are evenly spaced around the column);

- The number of pulses arriving at the column along all incoming edges will be summed over a short time period. If the column happens to have a connection from the pre–processor, the pulses arriving along that connection will also be added into the overall sum of arriving pulses, but with a higher weighting factor.

- Then a pulse train will be produced, whose instantaneous frequency $f$, called the *activity* of the column, is proportional to the value of the sum. This pulse train is further transmitted with the unchanged instantaneous frequency $f$ to the post–processor, if there such a connection, and with frequency $m_d f$ along each outcoming edge $d$.

The *memory characteristics* of a column refer to updates of communication "parameters" of a column:

- The mechanism, which sums the incoming pulses to the columns, will exhibit short term *habituation*. That is, if the sum is repeatedly large for a long time thus giving rise to a large value of $f$, then the mechanism will tire and begin reducing the value of $f$. Conversely, after a period of time when the sum is repeatedly low, the habituation or tiredness will slowly wear off. The habituation may only last a fraction of a second;

- for any edge $d$, whenever a pulse arrives from that direction, $m_d$ will be incremented slightly, provided that $f$ exceeds a certain value at the time the pulse arrives. The values of $m_d$ for all edges represent the memory stored in that column. The increased values of $m_d$ will persist for a long time, perhaps weeks, months or even (in the case of very large $m_d'$s) years, but they too will slowly decay over the time.

The function of each column, as described above, can be implemented in a variety of ways and the way of the implementation is not essential for further explanation. Nevertheless, for the sake of plausibility Goldschlager sketches column implementation with the help of a model of some variant of "spiking neurons" which, however, are not specified in sufficient details that would enable a more rigorous treatment of the proposed implementation.

## 5.3 Patterns

The knowledge representation in the memory surface is best viewed as a series of levels, similarly as in the case of the Valiant's previous model. Progressively higher levels deal with progressively more abstract concepts ("items", as we called them in part 4.6). At the lowest level, we find the most fundamental concepts, which will be represented by the *simple patterns*, which will be introduced in the sequel. Associations (see 5.5) between any two such simple patterns will represent the next level of concepts, etc.

Define a *pattern* P to be any set of columns, together with their relative activities. Many of these activities will be zero, representing the fact that the corresponding column does not participate in this particular pattern, and many will be non–zero, thus representing an important part of the pattern. The convention will be adopted that every pattern P is normalized, i.e., the sum of activities of all columns in the pattern is exactly one. When all columns in the pattern change their activity, as opposed to the previous activity, by a multiplicative factor $s$, for any non–negative $s$, it will be said that the pattern is currently *active with strength* $s$ on a memory surface.

In practice, each pattern will consist of many thousands of active columns. Each pattern will represent a concept that the brain can handle. These concepts may be less abstract such as "dog", or more abstract such as "ownership".

The simplest patterns which correspond to the least abstract or most fundamental concepts, which the brain can handle are just those patterns which result from some combination of receptors firing in response to some actual events occurring in the environment of the brain. Symmetrically, those patterns which cause effectors to produce some useful impact on the environment, are also among the simplest patterns.

## 5.4 Computing on the Memory Surface

Now we shall briefly describe the computational characteristics of the whole memory surface.

One important observation can be made immediately. At any time various columns will be active to various extents. So over any period of time, each column will experience some average amount of stimulation in the form of pulses arriving at the columns. It will therefore tend to habituate to this average stimulation, which may be thought of as *background noise.*

The computational characteristic of the memory surface may be considered in terms of the interaction of patterns.

The simplest case is when the pattern consists of only one column $A$, whose firing is independent of (i.e., un–correlated with) the firing of any other columns. Then, averaged over time and any edge $d$, $A$ will receive the same number of pulses per second along each edge $d$. Therefore, even though $m_d$ is only altered when $A$ is active, the value of $m_d$ will reflect the average background noise which strikes $A$ along the edge $d$. Whenever $A$ becomes active, say with instantaneous frequency $f$, it will transmit a pulse train of instantaneous frequency $fm_d$ along each outcoming edge $d$. As these pulses arrive at adjacent columns, they will be relayed onwards and continue in the same direction. Therefore, pulses will arrive at every column of the memory surface. The net effect will simply be to increase the background noise somewhat.

The next simplest case to consider is when the pattern P consists of two columns $A$ and $B$, whose firing is somewhat correlated. As before, the values of $m_d$ for each edge $d$ and for both columns $A$ and $B$ will reflect the average background noise arriving from the remainder of the memory surface along each edge $d$, subject to one exception: this will present the edge with the weight $m_{AB}$ connecting $A$ with $B$, and the edge with the weight $m_{BA}$ connecting B with $A$. Whenever $B$ is active, it will transmit pulse trains in all directions, in particular, along the edge $(B, A)$, and the same holds for $A$. Since $A$ and $B$ are correlated, they tend to be simultaneously active more often than if $A$ and $B$ were active at random. Both $m_{AB}$ and $m_{BA}$ will be increased above the value which only represents the background noise. In this way the correlation is "remembered".

The process described above presents the basic mechanism of how the columns, which are simultaneously active will learn of each other's correlations. In [7] it is explained how a correlation stops propagating itself among the columns (or patterns) that are not correlated.

## 5.5 Associations

With a the similar consideration, as above, one can show that if several patterns are simultaneously active on the memory surface, they will become *associated* together by exactly the same mechanism. In the previous subsection, this mechanism was responsible for causing two correlated columns to store a memory of the correlation. Namely, all columns, which find themselves "in–between" the simultaneous activity of two or more patterns (i.e., along the path from columns in one pattern to columns in the other pattern) will learn about the correlation.

Thus, patterns P and Q can only interact by being simultaneously active on the memory surface. In that case, P will form an association with Q to a degree depending on the strength of Q and on the duration for which they were simultaneously active. If later P is activated, it will tend to activate Q with a strength, which depends on the strength of P and on the degree of association of P to Q.

We see that the mechanism of establishing associations is "built–in" in the memory surface model and works automatically, by itself. Making further use of the same association mechanism, Goldschlager goes on in explaining how more abstract concepts are formed, namely by association of simpler concepts, which are related by contiguity in time or place, cause and effect, and by resemblance.

*Contiguity in place* refers to the fact that when objects are often observed to be physically together they become associated. The frequent simultaneous observation of the objects will cause the patterns on the memory surface, which represent these objects to be simultaneously active for a long total duration. This explains the forming of an association between these objects.

*Contiguity in time* refers to the fact that when events often occur in a sequence over time, those events will become associated. Examples are remembering a song and learning the alphabet. E.g., in the latter case, as soon as the concept (of a symbol) $A$ becomes active, its columns will tend to stimulate each other and thus the pattern will persist on a memory surface for some time. However, due to habituation, the strength of the pattern will decrease with the time. By the time the concept $B$ becomes active, it will be simultaneous with the weaker strength pattern $A$. Therefore an association of $A$ with $B$ will be formed by the standard association mechanism. And so on with the next members of the sequence. This, by the way, explains why sequences, which are learned in one direction, are hard to recall in the reverse direction.

*Cause and effect* is to be understood as a special case of sequence.

When two concepts share some common set of features, they *resemble* one another. Hence, due to the very nature of an associating mechanism, two different concepts with a common set of features will have associative links to this set of features, even if the patterns representing the two concepts were never present simultaneously. Clearly, at the same time, the set of common features of two or more concepts presents an abstraction, or a generalization of the concepts involved.

## 5.6 Higher Brain Functions

### Train of Thought

How does a brain progress from one thought to another? Here a mechanism, similar to one causing the association of members of some sequence, is in action. At any moment of time, some sets of patterns will be active having various strength on the memory surface. Each of these patterns will tend to excite all the patterns, to which it is associated, with strength proportional to its own strength and to the strength (weights) of the associative links. Meanwhile, the currently active pattern will tend to habituate and their strength die away.

So the memory surface will exhibit a *flow of activity* from concepts to associated concepts. At any time, events in the outside world can activate their corresponding patterns on the memory surface, and these, together with all other currently active patterns, will give rise to all their associative patterns, and so on ad infinitum. Thus, rather than a single train of thought, the memory surface exhibits simultaneous trains of thought, each one branching out to many associated concepts. However, the overall level of activity will be kept in check, so that the weaker associations may not become active at all.

The currently active patterns may be thought of as the contents of *short–term memory*, whereas the associative links between patterns may be considered to be the contents of a *long–term memory*. Forgetting in the short–term memory occurs fairly rapidly through habituation. In the long–term memory, forgetting can take place in two ways. Firstly, the weight of links outcoming from columns slowly decreases with time. Secondly, if the association $A \rightarrow B$ is currently stored in the long–term memory and a new association $A \rightarrow C$ is formed very strongly, then although $A \rightarrow B$ is still present, it may be effectively forgotten (or blocked) because each time $A$ is activated, $C$ will become activated with a great strength and thus prevent $B$ from being activated.

### Creativity and Sleep

Creativity consists of a lucky simultaneous activation of two patterns with many features in common. In this way a new, so–far "unknown", abstract concept will be invoked. The luck may be provided by the external environment, or by a random stimulation in sleep. Sleep has two survival phases, namely to help the memory of infrequently used concepts and associations, and to encourage creativity.

### Self and Consciousness

The concept of *self* is a more or less accurate model which the memory surface forms to distinguish the objects it can directly control from those it cannot.

*Consciousness* is a complex of different concepts that all relate to the idea of awareness.

*Awareness of environment* is simply the activation of the appropriate concepts, of the memory surface which model that aspect of the environment. *Self–awareness* refers to the fact that, in addition to the awareness concept, the concept of self is simultaneously active on the memory surface. Finally, *awareness on one's own train of thought*, often called *introspection*, means forming an association between the concept of self and the other concepts currently active on the memory surface.

*Conscious thoughts* can be regarded as those patterns which are active simultaneously with the concept of self, and whose associations are sufficiently strong to form an association with the concept of self. *Unconscious thoughts* are those which are active when the concept of self does not happen to be active, and those patterns where activity is too weak to form an association with the concept of self.

**Free Will**

People can never completely predict their own behaviour. According to Goldschlager, this would be possible for some hypothetical smart computer that, having "on line", the same inputs as human brain, and operating in exactly the same way, but faster, would reach the conclusion sooner than the brain. In this way the computer could predict the behaviour of a person, which would then appear, to this computer, as a person without free will. In this sense people do not have free will. Otherwise, to the extent that people cannot predict their own behaviour, they feel that they have free will.

Randomization can bring yet another twist to this analysis. Assume that people have some source of randomness in their brains (as Goldschlager's model in fact assumes — see the explanation of the creativity, that requires a random stimulation; it can be implemented in the way described in Section 6.4 where random connections among neurons are established). When the above smart computer has no access to the same randomness generator as the observed person, it will be never able to predict the behaviour of that person.

# 6 Brain as a Molecular Computer

## 6.1 Introduction

When compared to previous models the last computational model of brain we are about to describe is quite unusual, indeed. On one hand, it is based on ideas, that seem to be equally influenced by the known results about the brain and mind, from biology, psychology, and (bio)chemistry, and some fresh ideas from computer science. Its originality stems from the fact that, in order to achieve the required functionality, (that, of course, is the same as the intended functionality of other models) the author makes use of the radically different means that were used in all the previous models, and that go below the neuronal level — viz. molecular level. What emerges is a specific model of a brain that only registers *patches* of information, and these are stored in the form of association between primitive signals.

## 6.2 Basic Ideas

Like all the previous models, de Bruijn's model [3] also concentrates on the part of brain memory that is responsible for processing of preprocessed signals from peripherals or from some intermediate pre–processors. It is a model that essentially spans over all levels of the three–level brain architecture suggested in the Section 1, with various attention paid to the individual levels. It also aims at the explanation of consciousness and unconsciousness.

The model can be best seen as a two layer model: the first layer describes the computation at the level of individual cells, while the second at the level of cell assemblies. (Note that there is no correspondence between levels as defined in Section 1, and layers, as we shall use them in the sequel).

Thus, the brain in this setting can be seen as a spatially distributed parallel computer, where the role of processors is played by the cells of the brain. Each processor (cell) is equipped by an associative memory with a retrieval mechanism.

The lower layer in this model is the *thinking soup* that describes how memorization is performed by molecules *in individual cells.* The upper layer is the *roaming random set* model, which describes how the information is distributed *over the cells.*

The goal of the design is to achieve that the organization of the information processing is independent of the positions of molecules and cells. It should neither depend on the number of these, although in both cases it is clear that the more the better. Also, an addition of some fresh cells at arbitrary places might improve the performance. There is no objection against occasionally stirring the cells around.

To achieve these objectives, both levels represent items of knowledge in a redundant manner, in a way that is not tied to any particular location. Thus, in the lower layer a memory item is not represented by a single molecule but by a set of identical molecules in a cell; these molecules do not have fixed positions. Similarly, from the point of view of the upper layer, each memory item is recorded in a relatively small number of cells, and when the memory is later consulted, the exact position of those cells becomes irrelevant.

The operational view of the model is as follows: the pre–processors are broadcasting a potentially infinite sequence of primitive signals $p_1, p_2, \ldots$ during the whole life of the system. On the other hand, they also often broadcast a query in which they are giving a signal they broadcasted in the past and are asking for the signal which immediately followed. The two–layer machinery should be able to record the sequence and to answer those queries in a real–time manner. As one could expect the basic mechanism used for answering the queries is that of *associative retrieval.*

## 6.3 Lower Layer

### Inside a Cell: the Thinking Soup

In the following one should omit the idea of a hard–wared processor; instead, we shall consider the following wet–wared processor.

Imagine a bowl containing a mixture of chemical compounds. Many chemical reactions take place. The idea is that every neuron, and, more generally, every unicellular organism contains such a bowl of soup.

There is a great, but finite number of compounds $A, B, C, \ldots$ that can appear in the bowl. That is why the number of all reactions that can ever take place in the bowl is finite. In the simplest case such reactions are of the form $A + B \rightarrow C$ stating that compound $A$ and $B$ will give rise to compound $C$. This is not necessarily a faithful description of a chemical reaction; rather, it should describe the fact that adding $A$ and $B$ to the existing mixture in the bowl will produce $C$.

The bowl can then act as a gigantic parallel computer, with different $A$'s and $B$'s representing inputs to some operations, and $C$'s the outputs, and with the whole computation performed with the help of reactions of the above mentioned type. If one thinks of DNA–like molecules of length $k$ than the number of possible compounds may be of order $c^k$ with some constant $c > 1$. This "molecular hardware" is potentially available, but most of it is never used.

Operating with such a thinking soup requires inputs and outputs. Input do not necessarily need consist of chemical ingredients inserted into the bowl at the beginning of its computations. There may be other phenomena, like electric or mechanical signals, that initiate the production of particular compounds. Similarly, the outputs might include sensors that transform chemical information into other kind of signals. So, according to de Bruijn, an input and output should be called an active and passive *smelling.*

It is quite possible that thinking soup helps biological system to *think*, along with neural networks, but de Bruijn does not seem to be able to provide any plausible explanations of such a phenomenon. Rather, he provides an interesting proposal of how the associative memory can be handled in the thinking soup.

## Implementing Associative Retrieval

Add an input $p$ to the thinking soup, followed immediately by the signal $q$. These inputs generate compounds $A$ and $B$ in the soup. Let these compounds generate a third compound, $C$: $A + B \rightarrow C$. Assume that there are mechanisms that restrict this reaction to the case where $A$ is followed by $B$ and not vice versa, and that this is the only way how $C$ can appear in the bowl. Further, let the compound $C$ be quite stable, and let it survive even after $A$ and $B$ have vanished completely. Then the whole process $A + B \rightarrow C$ can be seen as an information storage and the respective reaction might be appropriately called a *storage reaction.*

In order to obtain memory retrieval, one should assume that there is a companion, the so–called *retrieval reaction* $A + C \rightarrow B$. It has the effect that if $p$ should ever reappear, producing $A$, then $C$ subsequently helps to produce $B$; the result is that $q$ is obtained as the output. Of course one has to assume that $A$ did not already trigger the retrieval reaction at the time of storage reaction: perhaps when $C$ was formed, $A$ was consumed entirely by this reaction, or was too weak to trigger the retrieval reaction.

Every brain cell may be able to record many thousands of different associations in this way.

## 6.4 Upper Layer

### Wizard's Poem

Now we shall concentrate on the description of how the information is stored in and retrieved from memory cells (remember that so far we have spoken about thinking soup *inside a single cell*).

To explain the corresponding principles de Bruijn uses the following metaphor: the wizard's poem:

> On a big island, called Radio Island, inhabited by very large number of radio amateurs, there is a central radio station (CRS). CRS, as well as all amateurs, can both receive and broadcast. The amateurs can have radio contact with CRS, but not with one another. CRS is always on the air, but the amateurs are playing with their radio stations only now and then. They tune at random, completely independently of each other, in the average once a week, but never for long. Usually they switch for about a minute and then switch off again. They have no names known to CRS and move freely around the island. The population of amateurs counts about $10^{10}$ of individuals, but every year a few percent drop off, either by death or by irreparable failure of their radio equipment.
>
> One day a ship from another island brings an old wizard who knows a wonderful poem of millions of lines, with each line different from the others. Since the wizard is to die soon, the CRS wants to save the poem for the future. But no one on Radio Island can read or write or record sound. The only possibility is human memory, but no one on Radio Island can learn such a long poem. Each of the amateurs can memorize at most a few hundred lines. What makes the thing even worse is that the limited memory of amateurs is unreliable, too: if anyone has memorized a line of a poem and is requested to quote it later, the probability that the quotation is completed error–free is not more than about 70 percents.
>
> How can Radio Island as a whole save the poem for the future? Can memorization be distributed over those inattentive, unreliable amateurs who cannot be even addressed personally?

De Bruijn offers the following solution. CRS memorizes the first line of the poem and then lets the wizard broadcast the whole poem at a speed of about four lines a minute. A little beep is given at the beginning and at the end of each line so as to enable the listeners to discard any incompletely received lines. The wizard can take arbitrarily long breaks, but each new broadcasting session has to start with the last line from the previous broadcasting.

It is assumed that all listeners are able to memorize all the short sequences of lines which they happened to pick up during the periods of a minute or so, during which they are tuned in, and that is in spite of the fact that they can see no logical connection between these fragments. Successive fragments may be a week apart, and this may represent gaps of more than ten thousand lines.

So each amateur knows a few hundred unrelated fragments of few lines each; not even remembering in which order he or she received the fragments.

Years later, long after the wizard's death, but long before the death of all amateurs, Radio Island still knows the poem. CRS is able to recover it in the following way. It starts to broadcast the first line. Among $10^6$ amateurs tuned at that moment, there will be a considerable number, possibly about like 40 (this can be rigorously proven under certain probabilistic assumptions), who had been listening and had also remembered the first two lines as the wizard recited them. These amateurs are assumed to react by sending the second line back to CRS. CRS compares these answers and selects the majority of equal lines as the true second line. CRS broadcasts that line, maybe somewhat later, possibly to quite different group of listeners, and gets the third line in return. This goes on, and the whole poem unfolds itself faultlessly, or at least with a very high probability of it unfolding itself faultlessly.

What makes it work is the fact that for an arbitrary choice of two moments it can be expected that there is a reasonable number of amateurs who had been attentive not only during both moments, but also during the periods following those two moments, long enough to handle two successive lines.

De Bruijn then justifies, by probabilistic analysis, the correctness and optimality of all figures chosen in the wizard's poem and thus solving it.

It is clear that one can also make use of the same principle in the case where several radio stations and several wizards will broadcast and recite their poems concurrently. One just has to assume that the amateurs, whenever they tune in, will select at random what station they will listen to.

### Roaming Random Set

As it is clear by now, the upper layer of de Bruijn's model is a translation of the metaphor of the wizard's poem. Instead of CRS, one has to take the pre–processors, or sensors, of the brain. Instead of $10^{10}$ amateurs one has to take $10^{10}$ memory cells. Instead od remembering pairs of consecutive lines of a poem, we have to consider the ability of a cell to store pairs of consecutive primitive signals in the thinking soup. Instead of $10^6$ attentive amateurs at each moment one has to consider what de Bruijn calls *active window*: this is the set of (brain) cells that are communicating with the pre and post–processors of the brain at any moment. This set is changing all the time (viz. the sequel), and the membership of cells in this set presents a random event. That is why de Bruijn also calls this set the *roaming random set*.

The roaming random set is thus the point of our attention. At the same time it is the basis of information storage and retrieval. Namely, the pre and post–processors have at any given moment access only to those cells that belong to the active window of that moment, and hence can store or retrieve information from those window cells only, using the mechanisms described in subsection 6.3.

To finish the translation from Radio Island to the brain context, one has to also translate the time scale. In the case of a brain, the cells must "tune in" for periods of about half a second (this is called a *decay period*), and the average time that the amateur is inattentive (about a week) is to be replaced by something of order of a few hours in order to get the same inattention factor of about $10^4$, as in the case of radio Island.

### Implementing the Active Window

In order for the brain to resemble the Radio Island, two questions must be answered. What is the mechanism that replaces radio transmissions, and how do the cells determine when to tune in and when to switch off?

For both purposes the same mechanism can be envisaged. Namely, not only are the neurons in the brain connected randomly via their dendrites, but moreover, the places where they "touch" each other can be also randomly set "on" or "off" for signal transmission in both directions. Some of the neurons are connected to pre–processing parts of the brain. From here, primitive signals are broadcasted along the connections that are at this very moment "on", essentially to a randomly chosen set of cells. This collection of cells is the active window of the moment. It is assumed that the topology of the underlying graph is such that the signals can arrive at any cell after only a few steps between cells.

## 6.5 Consciousness and Unconsciousness

These two notions will be understood as particular *modes of interaction* between pre and post–processors and the memory. This interaction takes place in the active window. The information flowing from pre–processors is immediately stored in the cells of the active window where it remains very easily available during the decay period. During that period the pre and post–processors might repeatedly recall the stored memory items — *"thoughts"*, and to *think* about them. This is called *reflection*. During the short decay period they can be recalled, and new thoughts can be formed *about* them and stored. All this has to be done within the decay period for otherwise the information is much harder to get.

In the course of the above mentioned process there may be a considerable traffic between pre–processors and the cells of the roaming random set, and between the latter and the post–processors. But it may very well happen that only a small part of the cells from the active window is involved in the reflection. De Bruijn calls this part the *conscious* part, and all the rest the *unconscious* part. There is no sharp borderline between the two. Conscious work is so difficult because of all the information processing involved in refection. Since it is so hard, it cannot handle more than a small portion of what goes on in the active window.

## 6.6 Short- and Long Term Memories, and Learning

The last matter that de Bruijn is concerned with are the questions related to short and long–term memory. In the model at hand the former is simpler to explain: it is the immediate memory accessible in the cells of roaming random set that represents the instantaneous active window. The long–term memory is to be looked for in the lower layer. It is both the *quality* of molecules, as well as their *quantity* in the thinking soup, and in cells, that give rise to a whole spectrum of long–term memories with various expected length of memorization. The respective mechanisms can be based, both, on increasing the concentration of related compounds in the thinking soup and on increasing the number of cells in which the same items are stored.

The aspects of long–term memorization are closely related to the learning aspects in the model at hand. Since our model handles only sequences of primitive signals learning basically relates only to memorization of respective items, and to performance improvement in their retrieval. Performance can be improved by repeated consultation of the same memory item in much the same way as people are doing when they want to remember a certain telephone number. Doing so the item gets into so many cells until it can eventually happen that certain items are present in whatever window, and can be even present in any reflection (i.e., in any conscious part of thinking).

Clearly, the result of learning is not only a matter of having exactly the same knowledge in larger quantities. It should also mean that new associations of different kinds are generated. Unfortunately, de Bruijn model as described in [3] does not explicitly deal with associations, and therefore nothing more specific about this can be thus far said. Nevertheless, we shall return to these and similar aspects in the next section.

For a lot of other details see the original paper. Do not be surprised that there are no references at all to other works.

# 7 Afterthoughts

## 7.1 Introduction

We have seen four computational models all of which can be seen as some computational approximations of a brain. In describing them we have purposefully postponed their mutual comparison. But we will do it now, having basic ideas about individual models, about their architecture, about their abilities to model certain cognitive tasks and, last but not not least, being armed with the respective terminology. As in any young discipline, this terminology is not yet quite unified. This is especially annoying in the case of such basic notions like concepts, memory items, patterns, etc. Nevertheless, for our further purposes we shall continue in the informal style of speaking about the models and depending on the model we shall preferably use the notions as introduced by their authors.

## 7.2 Comparing the Models

By mapping the respective parts (as far as they exist, and as long as this correspondence is plausible) of the previous four computational models one to an other, we get a good basis for their comparison.

Referring to the three level brain architecture mentioned in Section 1, the lowest level is modeled by (the first generation of) neural nets, neuroidal nets (in Valiant's approach), memory surface (Goldschlager), and net of cells (de Bruijn). Out of these four, the first two are formalized to such an extent that their computational equivalence can be shown by mutual simulations. This is mainly due to the fact that both nets are based on a notion of neuron or neuroid, respectively, that are quite similar from their functionality point of view. Because of this, as far as the brain models are concerned, we shall not make any distinction between these two models in the sequel.

The memory surface model is based on the notion of columns whose behaviour has not been described by the author in full detail. Goldschlager has sketched their implementation with the help of spiking neurons; these in turn can perhaps be simulated by neuroids, as discussed in Valiant [22] (the possibility of this simulation depends on the exact definition of spiking neurons at hand which was, unfortunately, not provided in the original paper [7]). Thus, it appears that the notion of memory surface can be formalized in a similar way as the previous two with the result that might be called *columnar net*, and all these three formalisms will be equivalent from computational point of view. Though, the result by Maass [11] mentioned at the end of subsection 3.4 points to the fact that spiking neurons can be much more economical (in terms of their total number) in performing certain tasks than the first generation neurons. Apparently, the relationship of spiking neurons to neuroids has not been so far investigated.

Apparently, de Bruijn's model, although based on other fundamental principles, can be also simulated by neural nets: each cell in his model will correspond to a neural net that shall be able to perform the required task of storing and associative retrieving of primitive signals. These nets will present the lower layer in de Bruijn's model. By connecting these "subnets" into a network, we can get the upper level. A further mechanism that would allow for random connections among the subnets will be necessary in order to simulate the roaming random set, but this seems to be still reasonable. The reverse simulation of neural nets within the de Bruijn's model can be imagined as well.

Thus, quite surprisingly, all of the four models seem to be interchangeable as far as their computing abilities are concerned. Some kind of a lower estimate on their computational power is provided by the studies of neuromata. The respective results state that they recognize exactly regular languages and as long as the nets are large enough, they can, in principle, realize any Boolean function with a bounded number of inputs very efficiently, and can perform associative retrieval if output capabilities are added.

What makes them unusual as computational means is their large descriptional complexity as compared with the length of inputs they process, their ability to learn, and their large degree of parallelism both at the input and processing levels.

Considering the learning capabilities of these devices we are shifting to the next level in the three level brain architecure. The respective features have been investigated in the most formal way, that is appropriate for computational treatment in the Valiant's model. The knowledge representation used here — namely that given by hierarchy of concepts, basically realized by Boolean circuits, seems to be close to that of Goldschlager. Although the comparison is little bit difficult since Goldschlager was not very specific as far as implementation details are concerned, his notion of hierarchy of patterns seem to correspond to the hierarchy of concepts in Valiant. However, the mechanisms for creating new concepts seem to be slightly different in both models. The main difference seems to lay in the fact that in Goldschlager's model associations tend to emerge automatically, be it by resemblance, contiguity in time or place, or by cause and effect. Such diversity has not been shown for Valiant's model (though other cognitive tasks not mentioned in Goldschlager have been considered by Valiant — cf. non–monotonic reasoning). Although it seems that all these kinds of associations are not completely out of the reach of Valiant's model, Goldschlager's solution seems to be remarkably uniform. It could be of interest to develop Goldschlager's model further by making it more formal. It is our guess that the two approaches could benefit much from each other and that they would also to a large extent appear equivalent.

In this comparison de Bruijn's model seems to be out of the competence since it does not deal with learning phenomena, except in a trivial manner. However, here one can also imagine how the model of storing and associative retrieving can be expanded in order also to include the creation of associations. To achieve this, one can postulate e.g. that the (bio)chemical storing and retrieving reactions inside cells (in the thinking soup) have spontaneous abilities to create molecules representing the associated information as derivatives of the respective compounds. The other possibility would be to delegate the learning of more complicated associations to the upper layer in much the same way as in Valiant's or Goldschlager's approach.

Thus, again, a remarkable coincidence of all three models seems to manifest itself. Based on this confluence, one can perhaps speculate what should be the "right" computational model of a brain at this medium level. A picture of some "algebra of thoughts" seems to emerge, with thoughts being represented by concepts, or tuples of attributes. There are many different attributes, and each concept is defined by only relatively few of them. Over these concepts, some set of basic operations is defined — the most fundamental ones seem to be the operation of storing and of associative retrieval (on the basis of partial match, say), and various kinds of associating operations. These should be based perhaps on resemblance (via common attributes), and on "followed–by", "occurring–simultaneously", and "cause–and–effect" properties of concepts as in Goldschlager's model. In parallel with such algebra one should also think about some kind of the corresponding logical calculus. Relational theory of databases can possibly serve as a kind of inspiration.

Finally, the upper level of higher brain functions is also handled in all three models. The most consistent theory seem to be that offered by Goldschlager and partially (but then in a more detail) also by Valiant. Assuming the functional equivalency of all three models, most of the results can be transported from one model to the remaining two. Only de Bruijn's model will need some special care — de Bruijn's explanation of consciousness and subsconsciousness seem to be different from the explanation of Goldschlager. But these two views do not appear to exclude each other and maybe both can be accommodated, especially in de Bruijn's model. By the way, it is interesting to observe what a role of importance is played by forgetting (habituation, and decay period) in both of the latter models. Some elements of randomness seem to be also essential in these models.

Not all higher brain functions have been explained, and the explanation does not always match the general opinion on the nature of the mental phenomenon at hand. One related question that seems to be passed by in all models mentioned here seems to be the question how people direct their train of though towards things that they want do do, but the corresponding stimuli are provided by no sensors. The answer in de Bruijn's model is perhaps hidden in the fact that, according to his theory, certain circumstances that are qualified by some mechanism (i.e., brain itself) as important can lead the brain to record the respective items with some urgent intensity. Therefore the respective concepts are present in every active window, and are present in every train of thought. The free will mechanism then selects the one of immediate priority and using the respective pattern the corresponding associations are invoked. These mechanisms can be then also implemented in other models. In fact, Valiant discusses what he calls *graded representation* of concepts that can serve the same purposes. In Goldschlager's model, the same can be achieved by keeping the respective pattern active with a high strength.

The last example can serve as an example illustrating the potential of the models at hand. It is an instructive *Gedankenexperiment* to try to devise an implementation of various other phenomena from the field of psychology not covered here, or to "transfer" some ideas on the implementation of higher brain functions from one model to an other. Only in this way the models can be adapted further, or refuted. But for a long time we will be probably not able to prove their "correctness" w.r.t. what happens in real brains. At best, we can hope that we shall be able to simulate the brain models on a computer and to observe (or model) some interesting mental phenomena. So let us see the prospects of such endeavours from the point of view of the current computer technology.

## 7.3 Simulating Models of a Brain

Let us make some quick calculations in order to get a feeling for what computational resources would be needed to realize computational models of a brain of a size comparable to that of a human brain. To get some meaningful insights, we shall make use of three different scenarios:

*Scenario 1 — Simplistic View:* For concreteness consider e.g. the Valiant's model because Valiant seems to be most definitive as far as the size estimates of his model are concerned.

The size of a human brain is estimated to be of the order of $10^{10}$ of neurons, and this seems to be a conservative estimate. Each neuron appears to have around $10^6$ to $10^8$ synapses, but take $10^4$ as a conservative estimate. The last figure compares favourably with Valiant's estimate of the size of a random graph underlying his model. Remember, in Subsection 4.6 we have estimated the optimal expected degree of an $N$ node graph as $\sqrt{N/r}$, with $r$ being a replication factor, of the order of 100, say. For $N = 10^{10}$ we find out that such a graph will have $10^{14}$ edges. This is at the same time an estimation on the number of floating point operations that would be required in order to simulate one (parallel) step of a brain (or of Valiant's model of it) on our computer. Assuming that the average firing frequency of a neuron is $10^2$ per second, we reach the conclusion that the brain appears to have a processing power of $10^{16}$ floating point operations per second. This is by factor of $10^3$ or $10^4$ more than the best current supercomputers can do.

But this is still only a part of the story. In order to represent the graph of a neuroidal net in the computer memory, we would need memory capacity of $10^{14} \approx 2^{49}$ (64 bit) words, that amounts to about $10^3$ terrabytes of memory. Roughly the same estimate would also hold for Goldschlager's model where columns would be realized by set of neurons.

To continue the story, we must now simulate a respective computation. Doing this in a sequential manner would lead to slowdown, as compared to real brain, of factor $10^3$, what is clearly unacceptable. Hence the only possible way out is to make use of a massive parallelism, but the respective computing capacity is probably not provided by considering all the existing supercomputers together. This is because in order to achieve a speed comparable to the brain we would need about $10^3$ supercomputers, when neglecting the communication complexity among them.

Thus, according to the previous analysis, it appears that unless we shall be witnessing some major breakthroughs in computing technology we shall be never able to simulate the human brain by current technology. As mentioned in [13], the computing power of current computers would be enough to simulate the neural capacity of a few simple creatures somewhere between a worm and a fly.

In this respect it is nevertheless appealing to observe that joint computational capacity of existing computer networks probably amounts to that required to simulate the human brain in the way sketched above.

*Scenario 2 — A More Detailed View:* Scenario 1 seems to suggest that in order to be able to meaningfully model (i.e., in an interesting way, from the computer science point of view) the human brain we shall have to wait until the computing technology will achieve such a degree of efficiency that would allow us to implement models according to the previous analysis. This line of reasoning is similar to what we hear, too often, in cases when some known solution is computationally inefficient. This is certainly not an answer that one would expect from computer science.

So let us try once again, this time with a closer look to de Bruijn's model. According to wizard's metaphor, in order to simulate the brain on Radio Island it was necessary that about $10^6$ amateurs tuned in at any moment. This points to the possibility that the whole brain does not have to be active at any time. Thus, returning to the computer realm, about $10^6$ processors of mega or gigabyte capacity would do for such purposes. This is quite a reasonable figure in the sense of a near future. It seems to be plausible that a processor and memory allocation schemes could be devised enabling to efficiently simulate a computational model of a brain with the parallel granularity and randomization similar to that as considered in de Bruijn's model. Similarly, the same line of reasoning can be adopted in the remaining two models. In Valiant's case, the cascades of neuron firings do not need to concern the whole brain, as well as in the Goldschlager's case, only the most active memory patterns must be taken into account. Moreover, in the case of neuroidal tabula rasa, similarly as in the case of hashing, not all the capacity of the underlying network is used. Some estimates from neurobiology and psychology judge that no more than 10% of brain capacity is ever used. This can also be a source of savings. The vital source of simplification can also perhaps stem from the fact that we are not after the faithful simulation of the brain, inclusively fear, pain, emotions, etc. If these properties will not appear in the underlying models spontaneously, somehow all by themselves, so to speak, then their omission could simplify the models.

Another avenue for computational resource savings is opened by considering other computationally more efficient neuron models, like spiking neurons, which, according to the result by Maass [11] mentioned in subsection 3.4, in some cases could replace hundreds of first or second generation neurons. The same may be true for Goldschlager's columns from section 5.2.

Speculations along similar lines bring us eventually to the third scenario.

*Scenario 3 — A Challenge to Computer Science:* In both previous scenarios we insisted on "machine" models of the brain which were inspired by biological neurons. From the engineering point of view, the machinery of real brains seems to be substantially different from that offered by current technology.

According to Parberry [13], current technology allows us (in 1994) to build circuits with:

- of the order of 100 inputs
- of the order of millions of processing elements
- these processing elements have only 2 inputs and compute very simple Boolean functions (like conjunction, disjunction, and negation).

Brain–like circuits appear to require

- of the order of $10^7$ inputs
- of the order of $10^{10}$ processing elements
- processing elements that have $10^3$ inputs, which are apparently computing complicated functions (directly mimicking neurons or neuroids)

Thus, simulating brain–like circuits (in fact neurons) by current circuits is a costly matter. But are the brain–like circuits, in brain–like computing, the only possibility? Cannot their use be omitted completely? After all, when designing an aircraft we do not mimic the birds their feathers inclusively. Why not try to devise such implementations of a brain that take advantage of the hardware as we know it today, i.e., of fast chips, and of reliable random access memories? Perhaps the design of a real brain has to take into account the factors and goals that are not necessarily to be considered in the case of our hardware: the slowness and unreliability of neurons, (and hence) the large degree of redundancy, the tolerance against physical injury and absence of some direct address mechanism.

Such an approach presents a gauntlet thrown to computer science, and to other related sciences that have to contribute as well. From the computer science point of view the crux lies in devising a *non–trivial machine independent model of a brain* (the medium level in our three level brain architecture mentioned in the Section 1). Only after having such a model can we start to think about its implementation on machines we have at our disposal. And these machines can be substantially different from our brains.

## 8 Conclusions

There is a trustable evidence that, since the end of nineteen forties, Turing wanted to build a brain [5]. His starting point was that mental processes are correctly described in the logical model, independently of the actual physical embodiment, and so can be embodied in a physical form other than the brain. According to Hodges [8], in order to recognize Turing's concurrent contribution to what is nowadays called Church–Turing thesis, we might more accurately enunciate a distinct Turing thesis with a somewhat different content: the Turing thesis is that *discrete–state–machine model is the relevant description of one aspect of the material world — namely the operation of brains.*

This is the basis on which the computational models of a brain, mentioned in this survey, are built. Despite their formal diversity, all the previously mentioned computational models of a brain show a remarkable confluence of ideas on how these models could look and what could be the main tasks performed by such models. At the same time they all support the validity of the Turing thesis in the form mentioned above.

Unfortunately we have no written evidence of what Turing's ideas were about his model of a brain. Turing probably did not arrive at a complete theory of what he meant by modeling the mental functions of the brain by a logical machine structure. Are we approaching the answer to this challenge by now?

**Acknowledgement&Apology.** I would like to thank Jan van Leeuwen for pointing me to the paper by de Bruijn [3]. Also, I apologize to authors of models referred to in this survey for sometimes perhaps too deliberate and/or imprecise interpretation of their ideas.

# Bibliography

[1] Arbib, M. A.: Brains, Machines, and Mathematics. Second Edition. Springer Verlag, New York, 1987, 202 p.

[2] Arbib, M. A. (Editor): The Handbook of Brain Theory and Neural Networks. The MIT Press, Cambridge — Massachusetts, London, England, 1995, 1118 p.

[3] de Bruijn, N.G.: A Model Of Information Processing in Human Memory and Consciousness. Nieuw Archief voor Wiskunde, Vierde serie Deel 12 No. 1–2 maart/juli 1994, pp. 35–48

[4] Churchland, P.S. — Sejnowski, T.J.: The Computational Brain. The MIT Press, Cambridge — Massachusetts, London, England, 1992, 544 p.

[5] Davis, M.: Mathematical Logic and the Origin of Modern Computers. In: The Universal Turing Machine: A Half–Century Survey, R. Herken (ed.), Springer–Verlag Wien, New York, 1994, pp. 149–174

[6] Gandy, R.: The Confluence of Ideas in 1936. In: The Universal Turing Machine: A Half–Century Survey, R. Herken (ed.), Springer–Verlag Wien, New York, 1994, pp. 55–112

[7] Goldschlager, L.G.: A Computational Theory of Higher Brain Function. Technical Report 233, April 1984, Basser Department of Computer Science, The University of Sydney, Australia, ISBN 0 909798 91 5

[8] Hodges, A.: Alan Turing and Turing Machine. In: The Universal Turing Machine: A Half–Century Survey, R. Herken (ed.), Springer–Verlag Wien, New York, 1994, pp. 1–13

[9] Indyk, P.: Optimal Simulation of Automata by Neural Nets. Proc. of the 12th Annual Symp. on Theoretical Aspects of Computer Science STACS'95, LNCS Vol. 900, pp. 337–348, 1995

[10] Kleene, S. C.: Representation of events in nerve nets and finite automata. In: C. Shannon and McCarthy, eds. *Automata Studies*, Princeton University Press, Princeton, NJ, 1956, pp. 3–41

[11] Maass, W.: Networks of Spiking Neurons: The Third Generation of Neural Network Models. NeuroCOLT Technical Report Series NC–TR–96–045, TU Graz, May 1996, 22 p.

[12] McCulloch, W. S. — Pitts, W. H.: A logical calculus of ideas immanent in nervous activity. Bull. of Math. Biophysics, 5:115, 1943

[13] Parberry, Ian: Circuit Complexity and Neural Networks. The MIT Press, Cambridge, Massachusetts, London, England, 1994, 270 p., ISBN 0–262–16148–6

[14] Perrin, D.: Finite Automata. In: Handbook of Theoretical Computer Science, Edited by J. van Leeuwen, Elsevier Science Publishers B.V., 1990, pp. 3–57

[15] Siegelmann, H.T.: Recurrent Neural Networks. In: Computer Science Today — Recent Trends and Developments (J. van Leeuwen, ed.), LNCS Vol. 1000, Springer Verlag, Berlin, 1995, pp. 29–45

[16] Siegelmann, H. T. — Sonntag, E.D.: Analog Computation via Neural Networks. *Theoretical Computer Science*, 131, 1994, pp. 331–360

[17] Siegelmann, H. T. — Sonntag, E.D.: On Computational Power of Neural Networks. *J. Comput. Syst. Sci.*, Vol. 50, No. 1, 1995, pp. 132–150

[18] Šíma, J.: Hopfield Languages. In: Current Trends in the Theory and Practice of Computer Science, Proceedings of the 22-nd seminar SOFSEM'95, Milovy, Czech Republic, LNCS Vol. 1012, Springer Verlag, 1995

[19] Šíma. J. — Wiedermann, J.: Neural Language Acceptors. In: Developments in Language Theory, Proc. of the Second International Conference, Magdeburg, June 1995, World Scientific Publishing Co., to appear

[20] Šíma, J. — Wiedermann, J.: Theory of Neuromata. ICS Technical Report 15/95, ICS AS CR Prague, submitted for publication

[21] Valiant, L.: Functionality in Neural Nets. Proc. 7th Nat. Conf. on Art. Intelligence, AAAI, Morgan Kaufmann, San Mateo, CA, 1988, pp. 629–634

[22] Valiant, L.G.: Circuits of the Mind. Oxford University Press, New York, Oxford, 1994, 237 p., ISBN 0–19–508936–X

[23] Wiedermann, J.: Complexity Issues in Discrete Neurocomputing. Neural Network World, 4, 1994, pp. 99–119

[24] Wiedermann, J.: Simulating Neuroidal Networks by Neural Networks. ICS Technical Report 17/95, ICS AS CR Prague, 1995

[25] Wiedermann, J.: Parallel Machine Models: How They Are and Where They Are Going. In: Proc. 22nd Seminar on Current Trends in Theory and Practice of Informatics SOFSEM'95, LNCS Vol. 1012, Springer Verlag, Berlin, 1995, pp. 1–30

[26] Wiedermann, J.: On the computational and descriptional complexity of finite neural networks (Invited Lecture). In: Preproceedings (Abstract of Papers), Workshop on Computability, Complexity and Logic WCCL'96, Zinnowitz/Usedom, March 1996, Preprint-Reihe Mathematik — Nr.1 — 1996, Ernst–Moritz–Arndt–Universität Greifswald, 1996, pp.17–19