Towards Machines That Can Think

Wiedermann, Jiří
1997

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Towards Machines That Can Think

Jiří Wiedermann

Technical report No. 715

September 1997

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic
phone: (+4202)6605 3520 fax: (=4202)8585789
e-mail: wieder@uivt.cas.cz

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Towards Machines That Can Think

Jiří Wiedermann[1]

Technical report No. 715
September 1997

Abstract

Recent progress in cognitive computing suggests that we might approach the point when the algorithmic principles of brain–like computing will be revealed and the study, design and realization of thinking machines will start to be an issue in computer science. For this purpose, we shall present a brief overview of related results from a machine oriented complexity theory.

Keywords

Cognitive Computing; Neural Nets; Molecular Computing; Brain-like Computing

[1]This research was supported by GA ČR Grant No. 201/95/0976 "HypercompleX" and partly by INCO–Copernicus Contract IP961095 ALTEC–KIT

1 Introduction

In 1950, in his seminal work Computing Machinery and Intelligence [20] A. M. Turing wrote: *I believe that in about fifty years' time it will be possible to program computers, with a storage capacity of about* 10^9, *to make them play an imitation game*[2] *so well that an average interrogator will not have more than 70% chance of making the right identification after five minutes of questioning. The original question, 'Can a machine think?' I believe to be too meaningless to deserve discussion. Nevertheless I believe that at the end of century the use of words and general educated opinion will have altered so much that one will be able to speak of machine thinking without expecting to be contradicted".*

While with his estimate of the storage capacity of today's computers Turing was astonishingly right, it appears that his further guess concerning the time when machine intelligence will become a reality was too optimistic. Sure, nowadays hardly anyone in computer science, artificial intelligence and related sciences doubts that in principle machines can think, in some form. At the same time, we do not know of any machine that would have passed Turing's test. Thus, the real issue is, whether at the present time machines can already think, and if yes, then how. I.e., do we have plausible arguments leading to the design of thinking machines?

It is the goal of this paper to present the known facts supporting the above arguments from the viewpoint of computer science.

In computer science the main approach to the problem of cognition, thinking, or intelligence in general, is to see these activities as a large scale interactive information processing phenomenon supported by a large scale computational machinery.

In understanding this complex task, nature has confronted us with different roadblocks, which in their variety and entirety, have not been challenged successfully either in computer science, 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 cognitive systems similar to the case of any abstract or formal computational system (cf. [29]).

At the highest "machine independent" level it is necessary to have some kind of functional specifications stating at a fairly high level (but as formally as possible, and as completely as possible) what kind of cognitive information–processing tasks will be performed and how they should interact. Informally and partially this is described as "higher brain functions" in other disciplines like cognitive psychology. It seems that intelligence cannot be captured entirely in the language of logic and that some non–logical calculus of interaction is required (see [26] for an interesting discussion).

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

Finally, one needs a plausible "machine" model of a brain in which the basic mind

[2] Turing's test for whether a machine can think: after a time limited conversation via a terminal one has to decide whether the dialogue was done with a man or a with a machine [20]

operations can be realized efficiently. Here again neurobiology and neuroanatomy, together with machine oriented complexity theory and artificial intelligence, can be of tremendous help.

No satisfactory specifications of any of the above mentioned three levels are known. It appears that a collaboration of all of the above mentioned branches of science are necessary needed in order to cope with the related problems[3]. At the border of all of these disciplines, a new discipline of computer science seems to have emerged that may be appropriately called *cognitive computing* [24].

Roughly speaking, each cognitive computational model, that we shall deal with in the sequel, will concentrate on a different level or on different levels with a various intensity of details, in the above mentioned sketch of a three level architecture of cognitive systems. In all cases there will be at least some loose reference to the remaining levels.

The survey is organized as follows.

In Section 2 we shall introduce a simplistic view of the brain as an interactive machine, or transducer. This view is common to all approaches included in this overview.

Then, in Section 3 we shall describe a family of the simplest models of a brain, viz. that of *finite neural nets*.

In the next four sections we shall present a model called *neuroidal tabula rasa* by Valiant [22], Goldschlager's model of *memory surface* from [7], then *the brain as a molecular computer* by de Bruijn [4] and we shall end with the recent model of *cogitoids* by the present author [30].

In Section 8 we shall briefly discuss anew the previous models in order to uncover their bottlenecks and superiorities and we shall finally try to answer the question whether thinking machines can be built at the present time.

As a suitable expository reading of results related to brains, machines, mathematics and computer science the books [1], [2], [6], and [14] are to be recommended.

2 Brains as Interactive Machines

The proper and the most general formal devices for describing cognitive activities seem to be *interactive machines* (cf. [26], [27]) whose task is to cope effectively with their environment by means of interaction. Incrementally they process infinite streams of input information supplied by the environment and produce similar streams of output information that is interpreted as the behaviour of machines. Since each increment to a previous input to such a machine may present a reaction of its environment to the machine's previous behaviour, such machines really allow for modeling interaction between them and their environment.

It is assumed that the streams of input data are produced by some preprocessors that, in practice, may themselves be a part of the brain. These preprocessors process

[3]In 1974, in his address [3] devoted to mathematical models of brains de Bruijn lists the following disciplines that could contribute to the solution of related problems: biology, psychology, animal psychology, psychiatry, parapsychology, pedagogy, sociology, history, archeology, medicine, pharmacology, physics, chemistry, philosophy, linguistics, engineering, computer science, mathematics

the data obtained by sensors into an encrypted, possibly simplified, form that is more suitable to be processed by the interactive machine itself. Similar situation also holds for output streams that are sent to postprocessors whose task is to decrypt data and instruct effectors appropriately.

By processing further parts of a potentially infinite input stream, the interactive machine is allowed to change its internal structure — so to speak to re–program itself repeatedly and to store some useful derived data — in order to learn from the past.

An important feature of our interactive model is that in many cases it cannot obtain the input data on its input ports "on demand" — rather, only when the data is available, when it is "offered" by the environment, or when it is spotted by the machine. Thus, the device works asynchronously with its environment and is at least partially input driven. Timing, i.e., data arrival time from individual preprocessors also plays a crucial role: it appears that in more elaborated models of cognition simultaneous or ensuing appearance of data is of vital importance for learning from experience.

By modeling the cognitive activities of the brain by using the interactive machine of the above mentioned kind, one in principle obtains a more powerful computational and descriptional tool than presented by standard Turing machines that only operate over inputs of finite length (cf. [27]).

3 Neural Nets

We shall begin our overview on models for cognitive computing with the simplest devices that are inspired by our ideas on how real biological brains work. The respective models are presented by various kinds of neural nets that differ in the realization of their various types of basic computational elements — artificial neurons.

In general, a neuron is a device that is able to compute values of a function $f : S^n \to T$ in a single computational step. Individual sorts of neurons differ from each other by the choice of function f (inclusively its domain and range). The elements $\vec{x} \in S^n$ are called inputs while the elements $y \in T$ are called outputs.

By using the outputs of some neurons in the place of inputs to other neurons we get a so–called *neural net*. The topology of the resulting net can be appropriately depicted with the help of an oriented graph in which neurons are its nodes, incoming edges denote the inputs to the neurons and outcoming edges denote the outputs from the neurons.

Any finite neural net can be seen as an interactive machine in the following way. Inputs that are not connected to the outputs of some neurons can be interpreted as *input ports* while outputs that are not connected to the inputs of some neurons present *output ports* of the net. All neurons in the net work synchronously. At the beginning of a computation the output values of all neurons are set to some prescribed values. Then, at each time step each neuron reads its inputs and computes its output according to its definition. Especially, at each computational step the net reads symbols, appearing at each input port, and sends one symbol to each output port.

Besides its topology the computational power of a neural net depends crucially on the nature of function f computed by the individual neurons. Therefore the computa-

tional power of each net must be investigated for each specific instance of f separately.

The simplest case of an artificial neuron is presented by the model by McCulloch and Pitts from 1943 [13]. A so–called *first generation neuron* is seen here as a highly idealized mathematical abstraction of real biological neurons, occurring in brains, that computes the values of the weighted threshold Boolean function.

Let $\vec{x} = (x_1, \ldots, x_n)$ and $\vec{w} = (w_1, \ldots, w_n)$ be integer vectors and t an integer constant. Then a Boolean function $f[\vec{w}, t]$ of n Boolean variables $x_1, \ldots, x_n$ is called a *weighted threshold* function if and only if

$$f[\vec{w}, t](\vec{x}) = \begin{cases} 1, & \sum_{i=1}^{n} w_i x_i \geq t \\ 0, & \sum_{i=1}^{n} w_i x_i < t \end{cases}$$

The vector $\vec{w}$ is called the vector of *weights* while t is called a *threshold*. It is said that a neuron *fires* if the sum of its overall stimuli, expressed by the weighted sum $\sum_{i=1}^{n} w_i x_i$, surpasses the threshold t.

In order to intuit the computational power and limits of finite neural nets of a first generation, the simplest case of single input port nets has been studied in the framework of formal languages and complexity theory. In the respective model, the input stream is read bit by bit and with a possible delay a stream of 1s or 0s indicating whether the input sequence, read so far, is accepted, is produced at the output. The respective nets are called *neuromata*. They have been studied intensively in works [18] and [19]. The respective results point to the fact that such devices have the computational power equivalent to that of finite automata — i.e., they recognize regular languages.

Second generation neurons (also called *analog neurons*) differ from the first generation by working over domains of real inputs and outputs. The biological motivation (that in practice seems to hold only partially) is that firing neurons produce electrical pulses with an intensity that varies in a certain range. This is modeled by the respective variables that 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* $\psi : R \to R$ 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 either be integers, or rationals, or reals), the computational power of the respective nets has a range rom 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 the most powerful computational devices that are know to us exist. To illustrate the computational efficiency of the respective networks, it can be shown that a finite number of analog neurons can simulate the 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. The above belief has recently obtained an unexpected support by the theory — in [11] it has been shown that the presence of arbitrarily small amounts of analog noise reduces the power of analog neural nets, and even the power of the the third generation (see in the sequel), to the power of the first generation networks (i.e., essentially to that of finite automata). This excludes the possibility that biological systems could possess the super–Turing computational capabilities.

Third generation neural nets are based on the model of *spiking neurons* that seem to better correspond to the computation of biological neural systems that employ pulses in order to transmit information among their computational units (i.e., neurons).

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. [10])

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

Thus, the neurons transmit information by encoding it into gaps among successive firings. For details see e.g. the paper by Maass [10] where the computational power of spiking neuron networks 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. A recent overview on the computational power of a pulse computation can be found in [12].

By devising and investigating increasingly more faithful models of biological neurons we are hopefully approaching the true computational efficiency of real brains. Unfortunately, unless some major computational feature of real neurons has been overlooked, it appears that no matter how faithfully we shall be able to simulate the activity of real neurons this effort is not going to substantially contribute to our quest for discovering the basic high level principles by which the mind works.

To summarize the computational power of neural nets from the point of view of complexity theory, the realistic models among them have equal power as the first generation nets. They differ merely in the efficiency of the respective devices: for the same computational task there may exist networks of a higher generation which are smaller than the networks of the lower generation.

In the framework of cognitive computing, the previous results can be interpreted as ones expressing that neural nets present efficient devices that can realize the basic cognitive task of pattern recognition, or an associative retrieval of them, with complexity characteristics, depending on the size and number of recognized patterns and on the computational power of individual neurons. The patterns are represented as finite segments of otherwise infinite input streams.

4 Neural Tabula Rasa

The main feature that seems to be missing in the previous neural net models is the ability to learn. The main obstacle, in this sense, seems to be the fact that these nets, once defined, cannot modify their computational behaviour. This appears to be a condition *sine qua non* for any learning device. This defect is eliminated in so–called *neuroidal nets* that present a programmable type of neural nets, which was first proposed by Valiant in 1988 [21]. The potential to model brain–like computations was further investigated in 1994 in the monograph [22], by the same author, and finally a neuroidal architecture for cognitive computation has been recently proposed in [25].

The main building element of neuroidal networks is *a neuroid* which is a combination of a neural threshold element (of the first generation) with the idea of a finite automaton and additional features that enable a neuroid to change its state, weights, and/or threshold depending on the activities of neighbouring connecting neuroids and current values of previous parameters. The idea that real neurons can store information not only in the synaptic strength (which is modeled by the respective weights) but also as *state* information is biologically plausible and experimentally confirmed [25].

The rules for changing respective parameters are formally described with the help of a *transition function*, similarly as it is the case of finite automata. readily, this enables the programming of individual neuroids (or more often: groups of neurons) to behave differently at different times.

E.g., one can design a single neuroid that, once "seeing" a certain input which contains at least one 1 will fire in the future iff it will be again confronted with a similar input. Such a similar input should have ones in the same places as the original input (and possibly in other places). Initially, the respective neuroid has all its weights set to 1. When confronted with an input consisting of $k \geq 1$ ones, it resets the weights at ports containing zero to 0 and its threshold to k. Then it enters into a "final" state in which no further parameter changes are performed any longer.

In this way one can program neuroids to implement various atomic cognitive tasks. Connecting neuroids in a net gives rise to a so–called *neuroidal tabula rasa* (NTR).

The most important task of NTR is its ability to represent various semantic *items* $x_1, \ldots, x_t$ which describe any aspect of the modeled reality. Each item is represented by a group of neuroids that fire under certain conditions on the inputs or internal computations corresponding to the moment when the NTR is somehow "reminded" of the item. The size of the group is called a *replication factor* and it must be chosen appropriately in order to ensure that the following NTR operations will work efficiently with a high probability (see the sequel).

By programming the neuroids accordingly, on its existing knowledge base of represented items the NTR is able to build further items hierarchically by using the following operations in any order:

1. *allocating* new neuroids to represent new items;
2. *memorizing* a new item $x_{t+1} = x_i \wedge x_j (1 \leq i < j \leq t)$ in the sense that x_{t+1} will fire at later times whenever both x_i and x_j fire (but not under other conditions);
3. *associating* x_i with x_j in the sense that x_i will fire whenever x_j fires;

4. *detect correlations* between pairs x_i, x_j that fire simultaneously;

5. do each of the above operations with relational or multi–object expressiveness rather than propositionally.

The respective operations are realized with the help of the so–called *vicinal algorithms*. They are probabilistic algorithms. Their most basic feature is that whenever communication is to be established between two items, not directly connected in the NTR, the algorithms establish the necessary connection via neuroids that are connected to both items. These neuroids are called *the frontier* of both items. Such a frontier, with the replication factor r, should exist with a high probability for any two randomly chosen groups of neuroids, with about the same replication factor r, representing two different items. This alone imposes a certain condition on the topology of the graph underlying the NTR.

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 nodes 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.

These two properties jointly are fulfilled e.g. in the class of random graphs on N nodes with the expected degree $\sqrt{N/r}$ [22].

For a detailed description of the respective vicinal algorithms see the monograph [22] where also the discussion of other cognitive tasks (e.g., memorization, correlational learning, relation expression learning, reasoning) can be found. Unfortunately, the resulting model of brain–like computations tends to be quite complex and it appears to be difficult to cope with e.g. higher brain functions more rigorously within this framework.

Further Valiant's elaboration of related issues in cognitive computing and the architecture of the respective systems can be found in [23], [24], and [25].

5 Memory Surface Model

Memory surface model by L. Goldschlager [7] appeared untimely in 1984 when the interest in cognitive computing in computer science was not as high as nowadays. From today's perspective this model seems to be of upmost interest since it provides the first coherent view on brain operations from a computational, or algorithmic point of view.

The model can be viewed as a directed graph comprised of millions of points, which will be called *columns*. The graph is to be seen in the three-dimensional Euclidean space and therefore it makes sense to speak about the distance of its nodes and the direction of its edges. Each column is connected to a number of 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. 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 and pre– and post–processors 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 the following rules:

- whenever a pulse arrives at the column along some edge it behaves as if it is travelling in a straight line through the column, providing that the edges are more or less 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 is such a connection, and with frequency $m_d f$ along each outcoming edge d.

- 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'_ds) years, but they too will slowly decay over time.

The function of each column, as described above, can be implemented in a variety of ways and the way of 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" (see Section 3) which, however, are not specified in sufficient details that would enable a more rigorous treatment of the proposed implementation.

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. 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.

The basic mechanism that causes various new and old patterns to emerge, with various activities, is that of the simultaneous excitation of the respective columns. Namely, when these columns are simultaneously excited by pulses coming from other active columns, or from pre–processors, to such a degree that they fire, the new patterns become active. Moreover, the outcoming pulses will cause the increase of weights of corresponding edges, memorizing thus the patterns with increased strength. Also, the pulses will arrive at post–processors causing some appropriate action of effectors. Due to their habituation, the patterns that caused the activation of new patterns will slowly decay while the newly activated ones will survive for a while.

In this way the memory surface works as an interactive machine.

The process just described is also responsible for memory surface learning of correlations among often simultaneously excited patterns. These correlations take the form of *associations* established among those patterns via edges with increased weight. New concepts emerge 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.

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 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.

With the help of the above mentioned mechanisms one can also explain the func-

tioning of higher brain functions.

For instance, 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 patterns will tend to habituate and their strength will die away.

So the memory surface will exhibit a *flow of activity* from concepts to associated concepts which we perceive as *train of thoughts*.

Creativity consists of a lucky, so far unprecedented 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.

The concept of *self* is a collection of various concepts that are all related to activities directly controlled by memory surface.

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.

6 Brain as a Molecular Computer

The next model that we are going to present originates from a series of papers by de Bruijn [3], [4], and [5]. After Turing, during the nineteen seventies de Bruijn was apparently one of the first pioneers in mathematics and computer science who paid serious attention to problems related to mathematical brain modeling.

His model is quite different from previous ones: it is not based on the preceding ideas of seeing the brain as a set of relatively simple, neuron–like computing elements connected by a fixed communication network.

The model can be seen as a gigantic spatially distributed parallel computer with tiny processors, equipped with sizable associative memories working at a molecular level, that only communicate with pre– and post–processors in a completely unpredictable way.

Within the framework of interactive machines, the operational view of the model is as follows: the pre–processors are broadcasting a potentially infinite stream 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 model should be able to record the stream 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*.

This mechanism is implemented in processors whose role is played by *cells*. Each cell is seen as a bowl containing a large, but fixed number of compounds $A, B, C, \ldots$. Hence, the number of reactions that can ever take place in the bowl is finite. In their simplest form the reactions are of the form $A + B \rightarrow C$. 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 huge parallel computer, with different As and Bs which represent inputs to some operations, and Cs the outputs, and with the whole computation performed with the help of reactions of the above mentioned type. This is why De Bruijn calls the respective mixture *a thinking soup*. If one thinks of DNA–like molecules of the length k, then the number of possible compounds may be of the order c^k with a 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. Inputs do not necessarily need to 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 kinds of signals. So, according to de Bruijn, an input and output should be called an active and passive *smelling*.

It is quite possible that a 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.

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 entirely consumed by this reaction, or was too weak to trigger the retrieval reaction.

Every brain cell and, indeed, any cell in principle (think of unicellular organisms) may be able to record many thousands of different associations in this way.

In order to explain how one could compute by using the thinking soup within cells, one has to return to the operational view of the model mentioned above. Assume preprocessors are continuously broadcasting a potentially infinite sequence of primitive signals. In the case that the respective signal has not been memorized yet in the cell, it gets memorized in the cell's thinking soup with the help of a storage reaction. Otherwise, the signal is interpreted as a query whose answer is retrieved with the help of retrieval operations and sent further to postprocessors.

Such an arrangement has a major drawback: the same information gets redundantly stored in each and every cell and therefore sooner or later the capacity of the systems will become exhausted. What is needed, is a mechanism that would store the items with significantly less redundancy in order to make better use of full storage capacity available in all cells.

To solve this problem de Bruijn offers an ingenious solution: at every moment not every cell is active and is ready to perform store or retrieve operations. The periods of activity of each cell are randomly chosen. Under such a scenario, signals only get stored into, or queries are only performed over, the set of currently active cells that is called *active window*. Since this is a continuously varying set deBruijn calls it "*roaming random set*". It is clear that by decreasing the size of the active window the storage capacity of the system increases while its retrieval reliability decreases. This is because some previously stored information does not need to reside in the cells of a current roaming random set and therefore a query asking for that particular information cannot be answered.

Here we are confronted with a nice optimization problem: what should be the size of a randomly chosen active window in order to guarantee, with a high probability, that there will be at least one cell from any window that was active in the past? The probabilistic analysis reveals that within the population of N cells the size of active window should be of order $\sqrt{N}$ [4] at any moment. Thus, most of the time, most of the cells can rest. For instance, considering 10^{12} cells in a human brain, it is enough that only about 10^6 from among them are active for the period of about half a second. This still leads to several hours of rest for each cell, on average, with an expected size of about 40 cells being in the intersection of a current active window and any past randomly chosen window. It is clear that with such a redundancy the system achieves a high level of fault tolerance. For more details, see the original paper [4].

To finish the description of de Bruijn's molecular computer two questions must be answered. What is the mechanism that implements broadcasting of signals among pre–processors, cells, and post–processors, and how do the cells determine when to be active and when to take a rest?

For both purposes the same mechanism can be envisaged. Namely, it is biologically plausible that not only are the neurons in the brain connected randomly via their dendrites, but moreover, the places where they "touch" each other can also be 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 the cells.

Besides the enormous storage capacity, de Bruijn sees the main advantage of his model in the fact that within its framework one is able to explain consciousness and unconsciousness.

These two notions are to 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.

For more arguments in favor of the above mentioned model see [3] and [5].

7 Cogitoids

Unlike in all of the previous cases, forget about the "machine" model of the brain now. Let us concentrate instead on the intended brain functions at the middle level of the three level architecture of a cognitive information processing system mentioned in Section 1.

Building on Goldschlager's ideas the respective approach was taken in [30]. The idea here was to consider the basic cognitive entities (concepts) and operations above them (such as forming of new concepts and associations among them) as the basic properties of the model. They are there and we do not care about how they are realized.

The resulting formal model of a *cogitoid* looks as follows.

The basic entity it handles are the *concepts*. They are modeled by subsets of a (large) finite universe $\mathcal{U} = \{f_1, f_2, \ldots, f_n\}$. The elements f_i of the universe are called *features*. Any object possessing every feature (and possibly some others) of a concept A is called an instance of A.

The concepts are denoted by upper case letters: A, B, etc. There is a special subset of concepts that is called *affects*. Affects correspond to positive or negative feelings of animals.

If A, B are two concepts and $A \subset B$, then A is called an *abstraction* of B and B is called a *concretization* of A. This is because any instance of A is at the same time an instance of B.

Especially, for any A and B $A \cup B$ is a concretization of either A or B while $A \cap B$ is their abstraction. Note that the set of all concepts over $\mathcal{U}$ with operations of set union and set intersection form a Boolean algebra with $\emptyset$ being its least element and $\mathcal{U}$ its greatest element.

With the help of the above mentioned two operations of concept union and intersection, new concepts can be formed from existing ones.

In a cogitoid, concepts may be explicitly related via associations. Associations emerge among concepts that occur in series or among similar concepts.[4] Formally, an ordered pair of form (A, B) of concepts is called an *association,* denoted also as $A \rightarrow B$. We say that A is associated with B. There are two types of associations: *excitatory* and *inhibitory.* Among any pair of concepts there may be both types of associations.

Two concepts A and B resemble each other in the concept C iff $A \cap B = C$ and $C \neq \emptyset$. This knowledge is also represented as an association $A \rightarrow B$.

At any time t any concept may be either *present* or *absent* in a cogitoid. If present, then a concept may be either in an *active* or in a *passive* state.

Also, at each time t there are two quantities assigned to each concept: its *strength* and its *quality.* The strength of a present concept is always a non–negative integer while absent concepts have the strength zero. The quality of concepts can be positive, negative, or undefined. Positive affects have always positive quality, while negative affects have always negative quality. The quality of other concepts may be arbitrary and depends on the history of concept formation or on the context in which a concept is invoked (activated).

Similarly, the strength is also assigned to each excitatory or inhibitory association.

Currently passive concepts may be activated either directly from the environment (via external stimuli that activate the corresponding features), or by internal stimuli via associations from other active concepts.

In the latter case, in order to activate, concepts should be sufficiently excited. The concepts get excited via associations. The strength of excitation depends on the strength and type of all associations leading from active concepts to the concept at hand. This concept is excited to the level that is proportional to the sum of strengths of all excitatory associations from currently active concepts decreased by the sum of strengths of all inhibitory associations from currently active concepts.

The cogitoid is seen as an interactive transducer that reads an infinite sequence of Boolean vectors of form $(x_1, x_2, \ldots, x_n)$. Each such a vector denotes an instance of a concept I that is determined by those features that correspond to ones in the above vector. The concept I represents an object that is "observed" by a cogitoid at its input.

The computation of C proceeds in rounds. At the end of each round a set of concepts is active. This set presents an output of the cogitoid — its behaviour, its reaction to the previous input. Let $\mathcal{A}_t$ be the set of concepts active at the end of the t–th computational round in a cogitoid C.

Each round consists of six phases:

Phase 1: *Producing the output and reading the input:* The concepts in $\mathcal{A}_t$ are sent to the output. All concepts in the set $\mathcal{I} = \{X | X \subseteq I\}$ corresponding to all abstrac-

[4]This is a slightly different presentation from that given in [30] where a resemblance of concepts was not represented explicitly by associations.

tions of I are activated. This models the formation of concepts by their simultaneous appearance.

Phase 2: *Activating new concepts by internal stimuli:* First a single new concept N from among all currently passive concepts gets activated. This is done with the help of a *selection mechanism* which inspects the excitation of all currently passive concepts from concepts in $\mathcal{I} \cup \mathcal{A}_t$ and subsequently activates the most excited concept N.

Simultaneously with activating N, also the set $\mathcal{N}$ of all abstractions of N gets activated.

Phase 3: *Assigning quality to concepts.* The quality of affects is constant all the time and it will determine the quality of all other currently active concepts to which active affects are related. The concepts whose quality cannot be determined by the preceding rule, get undefined quality.

Phase 4: *Long–term memorization:* The strength of all currently activated concepts is increased by a small amount.

Similarly, the strength of associations between each concept in the set $\mathcal{A}_t$ and each in $\mathcal{N}$ is increased. This models the emergence of associations by cause and effect.

Finally, the associations by resemblance are updated by increasing the strength of associations between each active concept in $\mathcal{I} \cup \mathcal{A}_t$ and each resembling present passive concept.

In the above mentioned process, if the association to be strengthened is between the concepts A and B, then if the quality of A was positive or negative or undefined, respectively, then the excitatory or inhibitory association, or both associations, respectively, between A and B are strengthened.

Note that increasing the strength of associations in some cases means that new associations are established (since until that time associations can be seen as those with strength zero).

Phase 5: *Gradual forgetting:* If positive, then the strength of all concepts that are not currently active and the strength of all associations among them is decreased by a small amount.

Phase 6: *Deactivation:* The concepts in the set $\mathcal{A}_t$ are deactivated and the set $\mathcal{N}$ becomes the set of all active concepts $\mathcal{A}_{t+1}$.

Note that the sequence $\{\mathcal{A}_t\}_{t \geq 0}$ models the "train of thoughts" in our cogitoid.

The notion of the above described cogitoid can be formalized with the help of sets and mappings.

In [30] it is shown that for any cogitoids it is possible to perform basic cognitive tasks such as abstraction formation, associative retrieval, causality learning, retrieval by causality, and similarity–based behaviour.

The next domain of behaviour that can be acquired by cogitoids is that of *Pavlovian conditioning.* This is a phenomenon in which an animal can be conditioned (learned) to activate a concept as a response to an apparently unrelated stimulating concept (cf. [22], p. 217).

For instance, one may first "train" a cogitoid to establish a strong association $S \rightarrow R$. Then, we may repeatedly confront such a cogitoid with a further, so far unseen concept A, with $A \cap S = \emptyset$, that is presented to it jointly with S, as $S \cup A$. After a while we shall observe that A alone will elicit the response R. Nevertheless,

after a few of such "cheating" from our side the cogitoid will abstain from eliciting R when seeing merely A (in psychology this is called *extinction*). Also more complicated instances of Pavlovian conditioning can be observed in *arbitrary* cogitoids. The only condition is that the cogitoids must be large enough to accommodate all the necessary concepts.

In order to explain Pavlovian conditioning no use of negative operant concepts and related inhibitory associations are necessary.

Cogitoids are also able to realize so–called *operant behaviour*. This is a behaviour acquired, shaped, and maintained by stimuli occurring *after* the responses rather than before. Thus, the invocation of a certain response concept R is confirmed as a "good one" (by invoking the positive operant concept P) or "bad one" (the negative operant concept N) only after R has been invoked. It is the reward (P), or punishment (N) that act to enhance the likelihood of R being re–invoked under similar circumstances as before.

The real problem here is hidden in the last statement which says that R should be re–invoked (or not re–invoked) only under similar circumstances as before. Thus, inhibition, or excitation of R must not depend on S alone: in some contexts, R should be inhibited, and in others, excited. Such a context is called an *operant context*; it is represented by a concept that appears invariantly as the part of the input of a cogitoid during the circumstances at hand. Thanks to cogitoid learning abilities, this operant context gets tied to the respective operant concept which, later on, causes that all associations emerging from this pair will inherit the quality of the operand concept at hand. Therefore, in the future, these associations will inhibit or excite R as necessary.

It appears that by a similar mechanism that ties a certain operant concept to some temporarily prevailing operant context one can also explain a more complicated case of the so–called *delayed reinforcing* when the reinforcing stimulus — a punishment or a reward — does not necessarily appear immediately after the step that will be reinforced.

All of the latter statements concerning the learning abilities of cogitoids can be formalized and rigorously proven (see the original paper [30]).

In the forthcoming paper [31] it is further shown in a kind of a thought experiment that when a cogitoid is exposed to a similar sequence of inputs as the human brain during its existence specific substructures start to develop in a cogitoid's memory. These structures correspond to episodic memories, frames for frequently performed activities, roles to be played by different objects in different contexts, attentional mechanisms for different contexts, and to habits. The latter represent the executive part of cogitoids memory. Within this framework language acquisition and generation can also be explained. Eventually, an emergence of behaviour, which resembles the behaviour which originates in the human mind, is to be expected.

8 Afterthoughts

We have seen several models of cognitive computing that to various extents can be thought of as certain computational approximations of abilities of human minds. It is

the ability of the cognitive models to algorithmically support or explain various mental phenomena as observed in experimental psychology that makes these models plausible.

It appears that from this point of view the simplest of these models — neural nets — can serve at best as a "cognitive substrate" for the implementation of more sophisticated models that deal with more advanced mental phenomena. Essentially, this was also the case with the molecular model that deals with a level of abstraction which is too low.

The addition of explicit learning abilities, which was the case with neuroidal nets, boosts the respective devices into a higher cognitive category. Nevertheless, the problem of a reasonable explanation of higher cognitive functions remains since the abstraction distance to be mastered, when speaking about their realization in terms of neuron firings, seems to be too great.

Goldschlager's model of memory surface seems to be the first one that liberated itself from a total dependence on a machine model and concentrated instead onto the mental entities — concepts — that are, as we believe, dealt with in our brains. This has immediately opened ways to more convincing computational theories of mind. It was only Goldschlager's insufficient level of formalization (and perhaps the absence of inhibitive mechanisms) that prevents in formulating and proving the respective theorems about cognitive abilities of the underlying model.

Therefore, thanks mainly to its mathematical treatment the model of cogitoids seems to be so far the only model that can pass the tests of explaining basic mental phenomena (such as Pavlovian conditioning or reinforcement learning) in a way that is usual in computer science. This model has provably a spontaneous ability of continuous learning. It presents thus a kind of universal learning machine that detects certain time–, space–, and similarity related patterns in the observed streams of data and learns the appropriate behaviour mostly by rehearsal and by punishment and rewards. Whether this model will also be able to cope with more complex mental phenomena remains to be seen. But now, already, one can see that this model with its calculus of concepts offers a firm explanation basis for such an enterprise [31].

9 Conclusions

Can machines think already at the present? An honest answer would be no, unfortunately, not yet, not in any interesting meaning of this word.

The good news is that we already have some clues of how the design of such machines should be approached, as we have seen in the paper.

The bad news is that even if we knew for sure how such machines should work, the next obstacle to be solved is their size. Namely, the human brain consists of about 10^{12} neurons and about 10^{15} dendrites. To simulate each neuron separately would require computers with memory capacities of that order. In order to keep pace with the brain one parallel computational step requiring an update of each neuron would call for realizing still a substantially larger number (than 10^{15}) of operations per second. No parallel computer can do that. Interestingly, the total memory and computational capacity of Internet seems to roughly correspond to the required task, if we forget about

the speed... Some models seem to be less computationally demanding than others: it is here where the true computational abilities of real biological neurons can play a significant practical role. If de Bruijn is right in his speculations that the processing and memory capacity of the brain is also hidden at the molecular level, then a long way is to be expected before we arrive at working models competing with human brains.

The next bad message is that when we insist on a human–like machine intelligence machine learning should apparently take the form of education similar to that which we undergo during our lives. Providing a machine with respective inputs can turn out to be an insurmountable technical problem. For machines the most suitable source of knowledge is that in a textual form. The question whether one can build cognitive machines educated purely with the help of textual material is therefore of upmost interest. Such a machine would then be inevitably crippled in its own perceiving experience and therefore could be hardly able to pass the Turing test. Nevertheless it can still be of enormous interest.

On the positive side, it can very well happen that we will discover ways of building and training true intelligent systems realized on computational models that are substantially different from neuron–based models and tailored to the possibilities of contemporary or foreseeable computing technologies. Even better (worse?), such systems might outperform our brains in cognitive abilities which is similar to the case that exists now in many application areas of computer science ...

The most important lesson to be taken from the above described models of cognitive computing is perhaps the following one: although the respective models appear to be formally different, in fact, from the view point of their cognitive abilities they are not. Higher level issues of more elaborated models can be simulated on simpler models. This hopefully points to a certain robustness of the respective class of interactive cognitive machines and supports the original Turing claim that mental processes can be described in some kind of logical model and that discrete state machines are a relevant description of brain operations [8].

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.: Mathematical Models for the Living Brain. Address to the Royal Netherlands Academy of Sciences and Letters, Section of Science, Amsterdam, 21 December 1974, with a postscript, added March 1975

[4] 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

[5] de Bruijn, N.G.: Can People Think? Journal of Consciousness Studies, Vol. 3, No. 5/6, 1996, p.425–447

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

[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] 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.

[11] Maass, W. — Orponen, P.: On the Effect of Analog Noise in Discrete– Time Analog Computing. Manuscript, 1996

[12] Maass, W. — Ruf, B.: On Computation with Pulses. A manuscript, 1997

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

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

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

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

[20] Turing, A.M.: Computing Machinery and Intelligence. *Mind*, Vol. **59**, 1950, p. 433–460

[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] Valiant, L.G.: Rationality. In: Proc. 8th Ann. Conference on Computational Learning Theory COLT'95, Santa Cruz, California, ACM Press, 1995, p. 3–14

[24] Valiant, L.G.: Cognitive Computation (Extended Abstract). Proc. 38th IEEE Symp. on Fond. of Comp. Sci., IEEE Press, 195, p. 2–3

[25] Valiant, L.G.: A Neuroidal Architecture for Cognitive Computation. Harvard University, Cambridge, MA, TR–11–96, November 1996, 29 pp.

[26] Wegner, P.: Tutorial Notes: Models and Paradigms of Interaction. See Peter Wegner's Home Page, 1995.

[27] Wegner, P.: Why Interaction is More Powerful Than Algorithms. Communication of the ACM, Vol. 40, No. 5, May 1997, p. 80–91

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

[29] 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

[30] Wiedermann, J.: The Cogitoid: A Computational Model of Mind. Technical Report No. V–685, Institute of Computer Science, September 1996, 17 pp.

[31] Wiedermann, J.: Towards Algorithmic Explanation of Mind Evolution and Functioning. In preparation, 1997.