On the Super-Turing Computational Power of Non-Uniform Families of Neuromata

Wiedermann, Jiří
2001

Institute of Computer Science
Academy of Sciences of the Czech Republic

On the Super-Turing Computational Power of Non-Uniform Families of Neuromata

Jiří Wiedermann

Technical report No. 849

December 2001

Pod Vodárenskou věží 2, 182 07 Prague 8, phone: (+4202) 6605 3520, fax: (+4202) 865 85 789,
e-mail:jiri.wiedermann@cs.cas.cz

Institute of Computer Science
Academy of Sciences of the Czech Republic

On the Super-Turing Computational Power of Non-Uniform Families of Neuromata

Jiří Wiedermann[1]

Technical report No. 849

December 2001

Abstract:

It is shown that the computational power of non-uniform infinite families of (discrete) neural nets reading their inputs sequentially (so-called neuromata), of polynomial size, equals to PSPACE/$poly$, and of logarithmic size to LOGSPACE/log. Thus, such families posses super-Turing computational power. From computational complexity point of view the above mentioned results rank the respective families of neuromata among the most powerful computational devices known today.

Keywords:
Neuromata, Turing machines with advice, non-uniform computational complexity, super-Turing computational power

[1]This research was partially supported by GA ČR Grant No. 201/00/1489

1 Introduction

The term 'neuromata' (a shorthand of 'neural automata') was introduced by Šíma and Wiedermann [5]. Neuromata differ from standard neural nets both in their functionality and in the way they receive their input. While standard discrete neural nets are usually considered as devices computing values of a Boolean function of a fixed number of input variables that are read by a net in parallel, neuromata act as recognizers of languages that read the respective words sequentially, via a single input neuron. Due to this fact a neuromaton can process input sequences of arbitrary, even of infinite, length. In [8] it was shown that neuromata also posses learning abilities.

The computational power and efficiency of a single neuromaton was studied in depth in [5]. One of the respective main results was that while the computational power of a single neuromaton equals to that of a finite automaton (and hence both recognize regular languages), the descriptional power of neuromata is much larger. There are instances of regular languages whose recognition requires exponentially larger finite automata as compared to the size of the respective neuromata. Thus, one can say that neuromata present a much more economical tool than finite automata for performing the respective tasks.

In this paper we will extend the studies initiated in [5]. We will study the power and efficiency of infinite families of neuromata. This is a standard approach in complexity theory allowing characterization of computational power of finite devices that are not capable to process inputs of arbitrary size. For the first time this approach has been used for determination of computational power of circuits whose size was bounded w.r.t. to the number of their inputs [3]. The respective results established the foundations of so–called non–uniform computational complexity theory. A similar framework has been also used for investigation of computational power of families of (discrete) neural nets of various kinds (for a recent overview, cf. [4]). A number of related results in neurocomputing also concerns the power of analog neural nets (cf. [2]) or of neuroidal nets [8]). Thus, a similar study of neuromata will complement these results. In our case, the problem is that the standard approach to infinite families cannot be applied directly since neuromata are not restricted to processing of fixed-size inputs. To overcame this problem the paper makes use of a recent idea from [7] that enables creating of infinite families also from devices (such as neuromata) that are capable to process inputs of arbitrary size. Thanks to this idea we will be able to characterize the computational power of neuromata in a novel way. As compared to families of recurrent neural networks the main result essentially shows that a different input convention of neuromata (that makes them functionally equivalent to finite automata) does not lead to degradation of computing power of the respective families which is equal to PSPACE/poly (for neuromata of polynomial size).

The structure of the underlying paper is as follows. In Section 2 we will briefly recall basic notions used in the sequel. We will start by reminding the definition of neuromata and we will proceed to the definition of infinite families of neuromata and of the languages recognized by such families. The main notions from non-uniform complexity theory are also mentioned. The main result characterizing the computational power of polynomially bounded families of neuromata will be given in Section 3. The proof of the respective result is constructive, giving direct simulation between neuromata and Turing machines with advice. Here also the consequences of our results and their comparison with other related results will be given. The merit of the respective results will be discussed in Section 4.

2 Preliminaries

Basically, a neuromaton is a restricted kind of a discrete recurrent (cyclic) neural net. We will give here merely an informal definition of a neuromaton. For a complete definition, see [5].

A *neuromaton* is a finite network of neurons. The topology of the network can be seen as a directed graph whose nodes are neurons and whose edges are directed connection among neurons. To each node and edge an integer number is assigned called the *threshold* and the *weight*, respectively. Each neuron is a discrete threshold unit that can be in an *active* or *passive* state. Signals are being sent along the directed edges. If an active neuron n_i sends a signal to its neighboring neuron n_j with threshold t_j along the corresponding directed edge (n_i, n_j) with weight $w_{i,j}$, then neuron n_i contributes to the excitation of n_j with the value $w_{i,j}$. If the sum of excitations of a neuron n_j from all its neighbors

(called a *total excitation*) achieves or exceeds the threshold value t_j then we say that n_j fires, i.e. it becomes active and sends a signal to its neighbors along its outgoing edges. If the total excitation of n_j remained bellow the threshold value n_j remains passive — no signal is sent.

There are two distinguished neurons to which no edges are entering, called the *input neuron* and the *terminal neuron*. There also is an other distinguished neuron called *output neuron* whose activity signals the acceptance of the input stream.

The *computation* of a neuromaton is defined as follows. Prior to the beginning of the computation, all neurons are set into prescribes states (passive or active ones). The terminal and the output neuron is set to the passive state. Then the computation begins. All neurons work synchronously. Let $x_1, x_2, x_3 \ldots, x_n$ be the finite sequence of Boolean inputs called the input stream. Then, at time $t = 1, 2, 3 \ldots, n$ the input neuron is set to the active state if and only if $x_t = 1$; otherwise it is set to the passive state. Each neuron computes its total excitation and depending on this value and the respective threshold the neuron enters the active state, or not. In such a way the computation proceeds until at time $t = n$ the input is exhausted. At that time the terminal neuron becomes active and remains so until the end of the computation. The computation then continues in the way as described before. If at some time $t \geq n$ the output neurons fires the input stream is accepted and the computation halts.

The language recognized by a neuromaton $\mathcal{N}$ is the set of finite input streams accepted by $\mathcal{N}$. The *size* of $\mathcal{N}$ is given by the number of its neurons.

In [5] it was shown that to each regular language R described by a regular expression of length m there is a neuromaton of size $O(m)$ that accepts R, and, vice versa, each neuromaton of size m accepts a regular language that can be described by a regular expression of size $O(m)$.

Now, in order to enable a consideration of more complex languages we will introduce the notion of infinite families of neuromata.

Let $\mathbf{F}_p = \{\mathcal{N}_1, \mathcal{N}_2, \ldots \mid size(\mathcal{N}_i) \leq p(i)\}$ be an infinite *family of neuromata* of increasing size bounded by polynomial $p = p(i)$. Such a family is also called a non-uniform family since in general there need not exist an algorithmic way to compute the description of $\mathcal{N}_i$, given i. Thus, there is no 'uniform' way to describe the members of the family. Intuitively, there is no better way to describe the family than to enumerate all its members.

The *language recognized by* $\mathbf{F}_p$ is the set

$$L(\mathbf{F}_p) = \bigcup_{n \geq 0} \{w | w \in \{0,1\}^n, w \text{ is accepted by } \mathcal{N}_n \in \mathbf{F}_p\}$$

Note that in order to define the above language we have restricted each neuromaton from $\mathbf{F}_p$ to work only on inputs of a fixed length. The class of languages accepted by families of neuromata of polynomial size will be denoted as POLY-NA. In addition to this class we will also consider the class LOG-NA (the class of languages recognized by neuromata of logarithmic size), the class POLY-C (languages recognized by the class of all Boolean circuits with a polynomial number of gates), the class POLY-NN (standard recurrent, or cyclic, discrete neural nets of polynomial size, reading their inputs in parallel), and the class POLY-FA (finite automata with a polynomial number of states).

Our main tool for characterizing the computational efficiency of the respective classes will be Turing machines with advice. Such machines have been introduced by Karp and Lipton in their seminal paper [3] establishing the foundation of non-uniform complexity theory. An advice is a special kind of an oracle. Effectively, an oracle allows inserting of outside information into the computation.) This information may depend on the concrete input and is given for free to the respective oracle machines. By this, the respective machines may gain the super-Turing computing power, viz. they can decide languages that are not recursively enumerable (cf.[1]). The difference between an oracle and an advice lies in the 'usefulness' of the additional external information. Contrary to the case of oracles, the advice value must not depend on a concrete input word; rather, it can only depend on the size of the input. Intuitively, the information delivered by an oracle makes sense only for the given input; the information offered by an advice can be used for all inputs of the same size.

Definition 2.1 *An advice function is a function $f : \mathbf{Z}^+ \to \Sigma^*$. An advice is called $S(n)$-bounded if for all n, the length of $f(n)$ is bounded by $S(n)$.*

Technically, a Turing machine with advice described by the advice function f operates on its input of size n in much the same like the standard Turing machines do. However, such machine can also call its advice by entering into a special query state. After doing so, the value of $f(n)$ will appear at the special read-only advice tape. Since that time the machine can also use in its computation the contents of this tape.

For the classes of languages recognized by Turing machines with advice, we will introduce a standard notation used for non-uniform complexity classes (cf. [1].)

Definition 2.2 *A language L belongs to complexity class $\mathcal{C}/\mathcal{F}$ iff L is recognized by a Turing machine from complexity class $\mathcal{C}$ with advice function $f \in \mathcal{F}$.*

Common choices considered for $\mathcal{C}$ that we shall use are: $LOGSPACE$ ('deterministic logarithmic space'), $PTIME$ ('deterministic polynomial time'), $PSPACE$ ('polynomial space'), etc. Common choice for $\mathcal{F}$ are *log*, the class of logarithmically bounded advice functions, and *poly*, the class of polynomially bounded advice functions.

For more information concerning the complexity of non-uniform computing cf.[1].

3 The power of infinite families of neuromata

Our next aim will be to show that the class POLY-NA coincides with the class of languages (standardly denoted as PSPACE/poly) accepted by polynomially space bounded Turing machines with a polynomially bounded advice function. We will do this by sketching a mutual simulation of neuromata by Turing machines with advice of the respective type.

Theorem 3.1 *The following assertions are equivalent:*

- $L \in$ *POLY-NA*
- $L \in$ *PSPACE/poly*

Proof: Let $L \in$ POLY-NA. This means that there is a polynomial p and a family $\mathbf{F}_p$ such that $L = L(\mathbf{F}_p)$. We will design a Turing machine $\mathcal{A}$ with a polynomial advice that accepts L in a polynomial space.

Define the advice function f of machine $\mathcal{A}$ as follows: for inputs of size n it assigns the description of the respective neuromaton $\mathcal{N}_n \in \mathbf{F}_p$. Since $size(\mathcal{N}_n) \leq p(n)$ there is a description of $\mathcal{N}_n$ that is also of a polynomial size w.r.t. n. Let $d(n)$ be the size of such description. Then, clearly, f is a polynomially bounded advice function. On input w, with $|w| = n$ machine $\mathcal{A}$ works as follows. It first calls its advice function (with argument n.). As a result of this call it gets on its advice tape the description of $\mathcal{N}_n$. This description is bounded by $d(n)$. Now all what remains to do is to simulate the actions of $\mathcal{N}_n$ on input w. Clearly, this can be done by $\mathcal{A}$ in space $O(d(n))$ (see the description of a computation of a neuromaton in the previous section). Thus, $L \in$ PSPACE/poly.

For proving the opposite direction we will have to develop a specific simulation algorithm that must reflect the fact that Turing machines and neuromata have different input mechanisms.

Let $L \in$ PSPACE/poly, let $\mathcal{A}$ be a single-tape Turing machine with advice, with a separate input tape. Let $\mathcal{A}$ accept L, let f be the respective advice function of size $d(n)$, let $\mathcal{A}$ be of space complexity $p(n)$. For each input of size n we will construct a specific neuromaton $\mathcal{N}_n$ that will accept exactly the words $w \in L$ of size n.

The size of the automaton will be bounded by $q(n) = O(\max\{n, d(n), p(n)\})$. It will store the current contents of both the advice and working tape of $\mathcal{A}$ and update it in accordance with $\mathcal{A}$'s action on input w. Note that only the part corresponding to the contents of the working tape has to be updated since the value of the advice function is given 'once for all times', for inputs of size n. The i-th cell on each tape will be represented by module M_i of neurons that in addition to the tape contents also represents the state of machine's finite control and the presence or non-presence of the respective tape head. The entities represented in module M_i include:

- the current symbol stored at i-th cell of the working tape; one neuron is needed firing iff the symbol stored is equal to 1;
- the i-th symbol of the advice tape (which is for all inputs of length n the same); one neuron will do, firing iff the respective bit of the advice is 1;
- the Boolean variable indicating the presence of the working head (one neuron);
- the Boolean variable indicating the presence of the advice head (one neuron);
- the current state of $\mathcal{A}$ (k neurons iff $\mathcal{A}$ has k states).

Thus, the values of the respective entities are represented by firing or non-firing of the respective neurons in M_i. The modules are concatenated to form a linear array enabling sending of signals to their neighbors, simulating in this way the movements of the heads on working and advice tape of $\mathcal{A}$.

There is one technical problem related to the fact that a neuromaton can only read its input via the single input neuron while a Turing machine with a separate input tape can copy any symbol of its input tape into arbitrary cell on its working tape. Moreover, a neuromaton must read its input at each step, until the input is exhausted, while a Turing machine can read its input at any time and can move freely over the input tape at any direction. In order to circumvent this problem we will force machine $\mathcal{A}$ to read all of its input w of size n, in one scan, and store it at the beginning of its working tape, in the first n cells. Since that time machine $\mathcal{A}$ will not be allowed to read its input tape — it will be forced to operate like a machine without the input tape, having the input written on the initial segment of its single working tape. All this can be arranged by 'reprogramming' the machine accordingly. What is important is the fact that this change will not affect the assumed polynomial space complexity of the underlying machine.

In order to achieve a similar distribution of input bits into the respective modules, our neuromaton first reads its input, bit by bit, via its input neuron that is located in the n-th module. After entering into the system at each time tick each input bit read keeps proceeding to the left neighboring module. This mode of activity is terminated when the terminal neuron announces to all modules to stop moving the bits to the left since the end of the input stream has been reached. At that time the i-th input bit finds itself in the i-th module as needed, for $i = 1, 2, \ldots, n$. From now on the neuromaton can start to directly simulate the actions of a 'reprogrammed' machine $\mathcal{A}$ which works as a single working tape machine.

The simulation is further complicated by the fact that the neuromaton has its working tape representation superposed on its advice tape representation. Thus, the simulation of each move of $\mathcal{A}$ requires the transfer of information between the head on the working tape and the head on the advice tape. Nevertheless, all this can be done, but the details are tedious and are left to the reader.

From this sketch of simulating neuromaton it is clear that it is of a polynomial size and that it can be constructed so as to accept exactly those words from L that are of size n. Thus, we can construct an infinite family $\mathbf{F}_q$ such that $L = L(\mathbf{F}_q)$. It follows that $L \in$ POLY-NA.

□

By a similar technique one can also prove an analogous result for neuromata of logarithmically bounded size. However, this time the use of a separate input tape of the respective Turing machine with advice must be restricted. We will assume that this input tape is one-way read only tape read in an on-line manner (that is, at each step the machine moves its input head) in order to compensate for the sequential reading mode of the corresponding neuromaton. The resulting claim, and the reformulation of the previous theorem is given in the following corollary.

Corollary 3.1 *(i) POLY-NA= PSPACE/poly and*

(ii) for an on-line one-way Turing machine, LOGSPACE-NA=LOGSPACE/log

It is known that POLY-FA=LOGSPACE/*poly* [7], POLY-C=PTIME/*poly* ([3]) and finally POLY-NN=PSPACE/*poly* (cf. [4] for numerous references). Referring to the known hierarchy of complexity classes (cf. [1]) and the respective separation results we get the following assertion:

Corollary 3.2 *POLY-FA* $\subset$ *POLY-NA, and*
LOGSPACE-NA $\subset$ *POLY-FA* $\subseteq$ *POLY-C* $\subseteq$ *POLY-NN* =*POLY-NA*

This brings an additional proof to that given in [5] that neuromata are more efficient than finite automata. On the other hand, when seen as non-uniform families the power of neuromata equals to that of general neural nets. The power of non-uniform circuit families lies somewhere in between that of finite and neural automata families.

4 Conclusions

We have shown that non-uniform families neuromata posses a super-Turing computational power. In general, it seems that non-uniformity jointly with interaction and infinite computations plays a much more important role not only in contemporary computing (such as represented e.g. by the Internet), but in general in all evolutionary interactive systems with a potentially endless life-span (such as human societies or artificial living systems in general - cf. [9]). This understanding has lead J. van Leeuwen jointly with the present author to the revision of the classical paradigm claiming that each computation can be captured by Turing machines. The extended paradigm states that *any computation can be captured by interactive Turing machine with advice* [6]. Thanks to their potential of processing potentially infinite input streams neuromata (enhanced perhaps by more complex output possibilities in the spirit of Moore or Mealy automata) can model interaction. The introduction of infinite families of neuromata allows the non-uniform computations. Thus, the infinite families of neuromata also support the extended paradigm mentioned above.

Bibliography

[1] Balcázar, J. L. — Díaz, J. — Gabarró, J.: Structural Complexity I. Second Edition, Springer, 1995, 208 pp.

[2] Siegelmann, H. T.: Computations Beyond the Turing Limit. *Science,* Vol. 268, April 1995, p. 545–548

[3] Karp, R.M. — Lipton, R.J.: Some connections between nonuniform and uniform complexity classes, in *Proc. 12th Annual ACM Symposium on the Theory of Computing* (STOC'80), 1980, pp. 302-309

[4] Orponen, P.: An overview of the computational power of recurrent neural networks. Proc. of the Finnish AI Conference (Espoo, Finland, August 2000), Vol. 3: "AI of Tomorrow", 89-96. Finnish AI Society, Vaasa, 2000.

[5] Šíma, J. — Wiedermann, J.: Theory of Neuromata. *Journal of the ACM,* Vol. 45, No. 1, 1998, pp. 155–178

[6] van Leeuwen, J. — J. Wiedermann: The Turing machine paradigm in contemporary computing, in: B. Enkquist and W. Schmidt (Eds), *Mathematics Unlimited - 2001 and Beyond*, Springer, Berlin, 2001, pp. 1139-1155.

[7] van Leeuwen, J. — Wiedermann, J.: Beyond the Turing Limit. In: B. Rovan, P. Ružička, Eds.: SOFSEM'01: Theory and Practice of Informatics. 28th Conference on Current Trends in Theory and Practice of Informatics, LNCS Vol. 2234, Springer, 2001, pp. 90–109

[8] Wiedermann, J.: The computational limits to the cognitive power of neuroidal tabula rasa, in: O. Watanabe and T. Yokomori (Eds.), *Algorithmic Learning Theory*, Proc. 10th International Conference (ALT'99), Lecture Notes in Artific. Intelligence, Vol. 1720, Springer Verlag, Berlin, 1999, pp. 63-76

[9] Wiedermann, J. — van Leeuwen, J.: Emergence of super-Turing computing power in artificial living systems, in: J. Kelemen (Ed.), *Artificial Life 2001*, Proceedings 6-th European Conference (ECAL 2001), Lecture Notes in Artificial Intelligence, Vol. 2159, Springer, Berlin, 2001