Fuzzy Neuroidal Nets and Recurrent Computations

Wiedermann, Jiří
2001

Institute of Computer Science
Academy of Sciences of the Czech Republic

Fuzzy Neuroidal Nets and Recurrent Fuzzy Computations

Jiří Wiedermann

Technical report No. 839

July 2001

Pod Vodárenskou věží 2, 182 07 Prague 8, phone: +4202 6605 3520, fax: (+4202) 858 57 89,
e-mail:jiri.wiedermann@cs.cas.cz

Institute of Computer Science
Academy of Sciences of the Czech Republic

Fuzzy Neuroidal Nets and Recurrent Fuzzy Computations

Jiří Wiedermann[1]

Technical report No. 839

July 2001

Abstract:

We define fuzzy neuroidal nets in a way that enables to relate their computations to computations of fuzzy Turing machines. Namely, we show that polynomially space-bounded computations of fuzzy Turing machines with a polynomial advice function are equivalent to the computations a polynomially-sized family of fuzzy neuroidal nets. The same holds for fuzzy neural nets which are a special case of fuzzy neuroidal nets. This result ranks discrete fuzzy neural nets among the most powerful computational devices known in the computational complexity theory.

Keywords:
Fuzzy computing, fuzzy neural nets, Turing machines, non-uniform complexity

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

1 Introduction

Fuzzy Turing machines as a formal means of describing and dealing with fuzzy algorithms were proposed by Zadeh [14] in late seventies. However, except of a very few immediate reactions (cf. [9], [5]) no much research was initiated by this notion. Recently, a revised model of fuzzy Turing machines corresponding to the rigorous foundations of fuzzy logic (cf. [2]) appeared in [13]. Here, fuzzy Turing machines corresponding to arbitrary fuzzy propositional calculus defined by a continuous t-norm have been defined and studied from the viewpoint of their computational power.

In parallel with the development of the notion of fuzzy Turing machines also the notion of fuzzy finite automata (cf. [9]) appeared. The latter model has recently received increasingly more attention. Namely, with the renaissance of neurocomputing, it appeared that, similarly as standard, 'crisp' finite automata (cf. [10]), also finite fuzzy finite automata can be efficiently implemented with the help of recurrent neural nets (cf. [6], [7]). In the latter mentioned paper its authors Omlin, Thornber and Lee Gilles raised a question whether computationally more powerful fuzzy systems (called *'recurrent fuzzy systems'* by the authors) than fuzzy finite automata could also be implemented with the help of recurrent neural nets. In this paper we will address the corresponding question within the framework of computations defined by the above mentioned recent model of fuzzy Turing machines which certainly represent 'recurrent fuzzy systems'. We define a class of discrete fuzzy neural nets in a way that neatly relate their computations to computations of fuzzy Turing machines. In order to obtain a result as general as possible we will consider nondeterministic versions of both fuzzy Turing machines and fuzzy neural nets. From the same reasons we will also consider more general nets than (discrete) neural ones — so-called neuroidal nets introduced originally by Valiant [11]. Neuroidal nets can be seen as programmable variants of standard neural nets.

As our main result we prove direct simulations between the class of fuzzy neuroidal nets and fuzzy Turing machines. Technically, we show that polynomially space-bounded computations of fuzzy Turing machines with a polynomial advice function are equivalent to the computations a polynomially-sized family of fuzzy neural nets. This result ranks discrete fuzzy neural nets among the most powerful computational devices known in the computational complexity theory. Thus, our result answers the previous question by Omline et al. affirmatively, at least for the case of discrete (recurrent) fuzzy neural nets.

The notion of a fuzzy Turing machine and that of its computations will be reminded in Section 2. In Section 3 a nonuniform version of fuzzy Turing machines — namely fuzzy Turing machines with advice, and the respective nonuniform complexity classes, will be introduced. In Section 4 the notion of fuzzy neural nets and that of infinite families of such nets will be defined. Next, in Section 5 our main result relating the computations of fuzzy Turing machines and fuzzy neural nets is proved. Finally, in Section 6 the the merits of our results, and some problems for further research are mentioned.

2 Fuzzy Turing machines

For the sake of completeness we briefly remind a basic variant of a fuzzy Turing machine as described in the original paper [13]. This model is obtained from the standard nondeterministic Turing machine by fuzzification of its 'instruction set'. However, the machine design is still general in the sense that the truth degrees associated with each accepting computation are computed by a special truth function called t–norm. Namely, this t–norm determines the kind of fuzzy propositional calculus in which the respective computations can also be described (cf. [12]). By choosing a particular t–norm we will obtain specific variants of fuzzy Turing machines, among them also the one defined originally by Santos [9]. As compared to this machine the acceptance criterion of the 'revised' machine from [13] is also changed so that the nondeterministic variant of the resulting machine satisfies Church-Turing Thesis (cf. [13] for the details).

Prior to giving the respective definition of a fuzzy Turing machine we introduce the definition of a *t–norm* (cf. [2]).

Definition 1 *A* t–norm *is a binary operation* $*$ *on [0,1] (i.e. t:* $[0,1]^2 \rightarrow [0,1]$*) satisfying the following conditions:*

1. $*$ *is commutative and associative, i.e for all* $x, y, z \in [0,1]$ *we have* $x * y = y * x$ *and* $(x * y) * z = x * (y * z)$;

2. $*$ *is non–decreasing in both arguments, i.e.* $x_1 \leq x_2$ *implies* $x_1 * y \leq x_2 * y$ *and* $y_1 \leq y_2$ *implies* $x * y_1 \leq x * y_2$;

3. *for all* $x \in [0,1]$ *we have* $1 * x = x$ *and* $0 * x = 0$.

(Note that we do not require a continuity of the t–norm.) For the sake of simplicity, we will introduce only the definition of a single–tape fuzzy Turing machine since we will be mainly interested in its principal computational abilities and less in its effectiveness.

Definition 2 *A* nondeterministic single–tape fuzzy Turing machine *(Fuzzy-NTM) is a ten–tuple* $\mathcal{F} = (S, T, I, \Delta, b, q_0, q_f, M, \mu, *)$ *where:*

1. S *is the finite set of* states*;*

2. T *is the finite set of* tape symbols*, to be printed on a tape that has a left-most cell but is unbounded to the right;*

3. I *is the set of* input symbols*;* $I \subseteq T$;

4. Δ *is the next-move relation which is a subset of* $S \times T \times S \times T \times \{-1, 0, 1\}$*. For each possible move of* $\mathcal{F}$ *there is an element* $\delta \in \Delta$ *with* $\delta = (s_1, t_1, s_2, t_2, d)$*. That is, if the current state is* s_1 *and the tape symbol scanned by the machine's head is* t_1*,* $\mathcal{F}$ *will enter the new state* s_2*, the new tape symbol* t_2 *will rewrite the previous symbol* t_1*, and the tape head will move in direction* d*. (In the previous relation symbol* -1 (1) *denotes a move by one cell to the left (right) and* 0 *denotes no move.)*

5. b*, in* $T - I$*, is the* blank*;*

6. q_0 *is the* initial state*;*

7. q_f *is the* final*, or* accepting *state;*

8. M *is a finite subset of the real interval [0,1], of cardinality* $|M| > 0$;

9. $\mu : \Delta \to M$ *is a function that to each move* δ *assigns the truth degree* $\mu(\delta)$ *of its membership in* Δ

10. $*$ *is a t–norm.*

Note that the membership degree of $\delta \in \Delta$ equals the truth degree of the proposition "δ *is an element of* Δ". For $\delta = (s_1, t_1, s_2, t_2, d) \in \Delta$ we will define a predicate $\Delta(s_1, t_1, s_2, t_2, d)$ and we will say that the truth degree of $\Delta(s_1, t_1, s_2, t_2, h)$ equals α if and only if $\mu(\delta) = \alpha$.

The notion of computation is defined as usual with the help of instantaneous descriptions (IDs). An *instantaneous description* Q_t of $\mathcal{F}$ working on input w at time $t > 0$ is a unique description of machine's tape, of its state and of the position of the machine's head after performing its t-th move on input w. If Q_t and Q_{t+1} are two IDs we will write $Q_t \vdash^{\alpha} Q_{t+1}$ and say that Q_{t+1} is *reachable in one step* from Q_t with truth degree α if and only if there is a possible move in Δ, with truth degree α, leading from Q_t to Q_{t+1}. On input w the machine starts its computation in the respective *initial* ID Q_0. This is an ID describing the tape holding a string of n input symbols (the so–called *input string*, or *input word*), one symbol per cell starting with the leftmost cell. All cells to the right of the input string are blank. The head is scanning the leftmost cell and the current state is q_0. From the initial ID the computation proceeds to IDs that are reachable in one step from Q_0, etc. If $Q_0 \vdash^{\alpha_0} Q_1 \vdash^{\alpha_1} Q_2 \ldots \vdash^{\alpha_{k-1}} Q_k$, with $\alpha_0, \ldots, \alpha_{k-1} \in M$ we say that Q_k is *reachable* from Q_0.

Now we establish a relation between the truth degrees of individual moves and those of achieving individual IDs. In order to do so note that, intuitively, within the propositional calculus belonging to the respective t–norm, for a particular move $\delta = (s_1, t_1, s_2, t_2, h) \in \Delta$ its membership degree $\alpha = \mu(\delta)$

can be interpreted as the truth degree of the proposition "*at each time t, when the machine is in state s_1, its head is scanning the i-th cell with symbol t_1, the machine will subsequently enter state s_2, rewrite the scanned symbol by t_2 and move its head in direction h*". Then, this truth degree can be extended to any ID of $\mathcal{F}$ reachable from its initial ID as follows.

Let Q_t be reachable from Q_0 via a computational path $Q_0 \vdash^{\alpha_0} Q_1 \vdash^{\alpha_1} Q_2 \ldots \vdash^{\alpha_{t-1}} Q_t$, with $\alpha_0, \ldots, \alpha_{t-1} \in M$, let $D((Q_0, Q_1, \ldots, Q_t))$ denote the truth degree of the proposition *"after t steps, starting from Q_0 and proceeding along the computational path $Q_0 \vdash^{\alpha_0} Q_1 \vdash^{\alpha_1} Q_2 \ldots \vdash^{\alpha_{t-1}} Q_t$, the instantaneous description of $\mathcal{F}$ is Q_t"*. The corresponding *evaluation function* D is defined as

$$D((Q_0, Q_1, \ldots, Q_t)) = \begin{cases} 1, & t = 0 \\ D((Q_0, Q_1, \ldots, Q_{t-1})) * \alpha_{t-1}, & t > 0 \end{cases}$$

Thus, the above mentioned truth degree is being 'adjusted' along any computational path with the help of the respective t–norm which acts as the truth function of the (strong) conjunction & (cf. [2]). Due to nondeterminism it may happen that Q_t is reachable from Q_0 via different computational paths. Therefore, the 'path independent' truth degree $d(Q_t)$ of the proposition "*after t steps, starting from Q_0 the ID of $\mathcal{F}$ is Q_t*" is defined as

$$d(Q_t) = \max\{D((Q_0, Q_1, \ldots, Q_t))\}$$

where the maximum is taken over all computational paths leading from Q_0 to Q_t.

A sequence $Q_0, Q_1, \ldots, Q_q$ of IDs is called an *accepting sequence of IDs* of $\mathcal{F}$ on input w, if and only if Q_0 is an initial ID, $Q_{i-1} \vdash^{\alpha_i} Q_i$ for $1 \leq i \leq q$, and Q_q is an accepting ID (i.e. such ID that contains the final state q_f). If an accepting ID Q_q is reachable from q_0 on input w we say that w is accepted with truth degree $d(Q_q)$.

Now we are in a position to define the acceptance criterion for a fuzzy Turing machine. A Fuzzy-NTM works as a (fuzzy) language acceptor as follows. The tape symbols of the machine include the alphabet of the language, called the input symbols, a special symbol *blank*, denoted b, and perhaps other symbols.

Definition 3 *Let $\mathcal{F} = (S, T, I, \Delta, b, q_0, q_f, M, \mu, *)$ be a Fuzzy-NTM. The input string w is* accepted with truth degree $e(w)$ *by $\mathcal{F}$ if and only if*

- *on input w, the computation of $\mathcal{F}$ reaches an accepting ID Q_k at depth k and $e(w) = d(Q_k)$;*
- *evaluation $d(Q_k)$ is not less than the evaluations of all IDs at the same depth*
- *k is the minimal depth at which such accepting ID exists.*

The *space complexity* of an accepting computation of $\mathcal{F}$ on inputs of size n is defined as usually, i.e. as a maximal amount of tape cells rewritten by $\mathcal{F}$ when processing inputs of that size.

Next we proceed to the definition of fuzzy languages. The idea is to define a fuzzy language as a fuzzy set of words. That is, each word from the respective language belongs to it with a certain *membership grade* which is a real number between 0 and 1. If such a language is recognized by a fuzzy Turing machines then the membership grade of each accepted word is equal to its acceptance truth degree.

Definition 4 *The* fuzzy language accepted *by $\mathcal{F}$ is the fuzzy set of ordered pairs*

$$L_{\mathcal{F}} = \{(w, e(w)) |\ w \text{ is accepted by } \mathcal{F} \text{ with truth degree } e(w)\}$$

In the context mentioned above function e is also called *membership function* of $L_{\mathcal{F}}$. Observe that in Definition 4 only words with the maximal acceptance degree are considered. It is important to realize that $\mathcal{F}$ does not 'print' the truth degree $e(w)$ corresponding to the accepted word w. In fact, this would also be principally impossible due to the fact that truth degrees are real numbers. Rather, this truth degree is *only defined* by Definition 3, but *not explicitly computed* by $\mathcal{F}$.

Note that in case when $\mu(\delta) = 1$ for all $\delta \in \Delta$ the Fuzzy-NTM equals the classical acceptance criterion the standard NTM as defined e.g. in [3]. Such a machine is also called a *crisp Turing machine*.

3 Fuzzy Turing machines with advice

The ultimate goal of this paper is the design and study of fuzzy neuroidal nets. Our main tool for characterizing their computational efficiency will be fuzzy Turing machines with advice which we define in this section. Crisp Turing machines with advice have been introduced by Karp and Lipton in their seminal paper [4] 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. In order to prevent delivering of too much information in a single advice value (e.g. the results of all computations on inputs of size n) one usually considers bounded advice functions.

Definition 5 *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 *fuzzy Turing machine with advice* described by the advice function f operates on its input of size n in much the same like the standard fuzzy Turing machines does. 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 fuzzy Turing machines with advice, we will introduce a notation similar to that used for crisp non-uniform complexity classes (cf. [1].)

Definition 6 *The class $Fuzzy - \mathcal{C}/\mathcal{F}$ consists of fuzzy languages L for which there exists a $L_1 \in Fuzzy - \mathcal{C}$ and a $f \in \mathcal{F}$ such that the following holds for all n and inputs x of length $n : x \in L$ if and only if $\langle x, f(n)\rangle \in L_1$.*

Thus, a fuzzy language $L \in Fuzzy - \mathcal{C}/\mathcal{F}$ iff L is recognized by a fuzzy Turing machine from complexity class $Fuzzy - \mathcal{C}$ with advice function $f \in \mathcal{F}$. A common choice for $Fuzzy - \mathcal{C}$ that we will also use is $Fuzzy - PSPACE$ ('fuzzy deterministic polynomial space'). A common choice for $\mathcal{F}$ is *poly*, the class of polynomially bounded advice functions.

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

4 Fuzzy Neuroidal Nets

Similarly as fuzzy Turing machines from Definition 2 which were defined as standard Turing machines with fuzzy transition relations, fuzzy neuroidal nets will be defined as standard discrete neuroidal nets in which the behavior of neurons will be governed by a fuzzy parameter-update transition rule. At its end this approach will enable assigning of truth degrees to configurations of a neuroidal net at hand and eventually to speak about the truth degree of acceptance similarly as it was the case with fuzzy Turing machines.

In the sequel, in order to be compatible with our notion of nondeterministic fuzzy Turing machines we define a corresponding notion of nondeterministic fuzzy neuroidal nets, essentially making use of the original Valiant's definition (and notation) of discrete neuroidal nets [11].

Definition 7 An n–input nondeterministic fuzzy neuroidal net $\mathcal{N}_n$ *is an ten–tuple* $\mathcal{N}_n = (\mathrm{G}, \mathrm{I}, o, \mathrm{W}, \mathrm{Q}, \mathrm{T}, \Delta, \mathrm{M}, \mu, *)$, *where*

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

2. $\mathrm{I} \subseteq \mathrm{V}$, *with* $|\mathrm{I}| = n$ *is a distinguished set of* input neurons*;*

3. $o \in \mathrm{V}$, $o \notin \mathrm{I}$ *is a distinguished* output neuron*;*

4. $\mathrm{W} \subset \mathbb{Z}$ *is the finite set of integers called* weights*. To each edge* $(i, j) \in \mathrm{E}$ *there is a value* $w_{i,j} \in \mathbb{W}$ *assigned at each instant of time.*

5. Q, *with* $|\mathrm{Q}| \geq 2$ *is a finite set of the* states *of neuroids which a neuroid can be in each instant;* Q *consists of two kinds of states called* firing *and* quiescent *states. To each node* i *there is also a Boolean variable* f_i *called* activity *to have value one or zero depending on whether the node* i *is in a firing state or not.*

6. $\mathrm{T} \subset \mathbb{Z}$ *is a finite set of integers called* thresholds *of the neuroid. The elements of sets* Q, T, W, *and* f_i*'s are called* parameters *of net* $\mathcal{N}_n$.

7. Δ *is the* parameter update relation *which is a subset of* $\mathrm{Q} \times \mathrm{T} \times \mathrm{W} \times \mathbb{Z} \times \{0, 1\} \times \mathrm{Q} \times \mathrm{T} \times \mathrm{W}$. *Each element* $\delta \in \Delta$ *defines for each combination of parameters holding at time* t *a set of new values of the respective parameters holding at time* $t + 1$ *in the following way.*

   Let $\mathbb{Z}$ *be the set of all integers, let* $w_i \in \mathbb{Z}$ *be the sum of those weights* w_{ki} *of neuroid* i *that are on edges* (k, i) *coming from neuroids which are currently firing, i.e., formally* $w_i = \sum_{\substack{k \text{ firing} \\ (k,i) \in \mathrm{E}}} w_{ki} = \sum_{\substack{j \\ (j,i) \in \mathrm{E}}} f_j w_{ji}$*. The value of* w_i *is called the* excitation *of* i *at that time.*

   If $\delta = (q_i, t_i, w_{ji}, w_i, f_j, q'_i, t'_i, w'_{ji})$ *is an update from* Δ *and at time* t *neuron* i *finds itself in state* q_i*, has threshold* t_i*, carries weight* w_{ji} *at some edge, its excitation is* w_i*, and activity of neuron* j *is* f_j*, then at time* $t + 1$ *neuron* i *will transit into state* q'_i*, change its threshold into* t'_i*, and its weight* w_{ji} *into* w'_{ji}*;*

8. M *is a finite subset of the real interval* $[0, 1]$*, of cardinality* $|\mathrm{M}| > 0$;

9. $\mu : \Delta \to \mathrm{M}$ *is a function that to each update* δ *assigns the truth degree* $\mu(\delta)$ *of its membership in* Δ

10. $*$ *is a t–norm.*

An *instantaneous description* (ID) of $\mathcal{N}_n$ at time t is a list of states and thresholds of all neurons followed by a list of weights of all edges in $\mathcal{N}_n$ at that time. The respective lists of parameters are pertinent to neuroids ordered by their labels and to edges ordered lexicografically w.r.t. the pair of labels (of neuroids) that identify the edge at hand. Thus at any time an ID is an element from $\{\mathrm{Q} \times \mathrm{T}\}^{|\mathrm{V}|} \times \mathrm{W}^{|\mathrm{E}|}$.

The *computation of a neuroidal network* is determined by the *input* and the *initial conditions*. The input is a sequence of n Boolean values which specifies the activities of input neurons within set I at time $t = 0$. Thus, the respective input neuroids are forced to fire or are prevented from firing at that time by mechanisms outside the net (by peripherals). The initial conditions specify the initial values of weights, states and threshold of all neuroids. These are represented by the initial ID. Of course, the initial conditions must be compatible with the input.

A *computational step* of neuroidal net $\mathcal{N}_n$ which finds itself in an ID Q_t is performed as follows. First, all excitations w_i are computed for this ID in parallel. Then parameter updates are realized for each neuroid i in parallel, in accordance with the respective update relation Δ. If for a neuroid there are several possibilities for performing an update then one of them is nondeterministically selected and realized. In this way a new configuration Q_{t+1} is entered.

The result of the computation after the t–th step is the $|\mathrm{V}|$–tuple of states of all neuroids in Q_{t+1}. This $|\mathrm{V}|$–tuple is also called the *action* at time t. Obviously, any action is an element in $Q^{|\mathrm{V}|}$. Then the next computational step can begin. We say that a computation of $\mathcal{N}_n$ on input w, with $|w| = n$ *terminates at time* t if and only if for some t we have $Q_t = Q_{t+1}$ and the output neuron o of $\mathcal{N}_n$ fires at that time. Q_t is then called a *terminating ID.*

Similarly as in the case of fuzzy Turing machines we establish now a relation between truth degrees of individual updates performed by $\mathcal{N}_n$ and the respective IDs. For the initial ID Q_0 we define its

truth degree $D((Q_0)) = 1$. Let $D((Q_0, Q_1, \ldots, Q_t))$ be the truth degree of the statement *'starting from Q_0 and proceeding via Q_1, Q_2, etc., after t steps the ID of $\mathcal{N}_n$ is Q_t'*. If $\alpha_1, \ldots, \alpha_{|\mathrm{V}|}$, respectively, are truth degrees of updates performed by neurons $1, 2, \ldots, |\mathrm{V}|$, respectively, in the $(t+1)$–st step, then $D((Q_0, Q_1, \ldots, Q_{t+1})) = D((Q_0, Q_1, \ldots, Q_t)) * \alpha_1 * \ldots * \alpha_{|\mathrm{V}|}$. With the help of the latter definition we can define the truth degree $d(Q_t)$ of reaching Q_t from Q_0 irrespective of the computational path by which Q_t has been reached as follows. We define

$$d(Q_t) = \max\{D((Q_0, Q_1, \ldots, Q_t))\}$$

where the maximum is taken over all computational paths leading from Q_0 to Q_t. Now we can proceed to the notion of acceptance by fuzzy neuroidal nets and of a language recognized by fuzzy neuroidal nets.

Definition 8 *Let $\mathcal{N}_n = (\mathrm{G}, \mathrm{W}, \mathrm{Q}, \mathrm{T}, \Delta, \mathrm{M}, \mu, *)$ be a nondeterministic fuzzy neuroidal net. The input string w, with $|w| = n$ is* accepted with truth degree $e(w)$ *by $\mathcal{N}_n$ if and only if*

- *on input w, the computation of $\mathcal{N}_n$ reaches an accepting ID Q_k after k computational steps and $e(w) = d(Q_k)$;*
- *evaluation $d(Q_k)$ is not less than the evaluations of all IDs reachable after k steps;*
- *k is the minimal number of steps in which such accepting ID can be reached.*

Note the compatibility of the above acceptance criterion with that of fuzzy Turing machines from Definition 3. In order to define a fuzzy language accepted by neuroidal nets we have to define an infinite family of neuroidal nets, having one member for each size of inputs. Moreover, we will require that each member of such a family 'works' over the same set M of possible truth degrees, and with the same t–norm. This leads to the following definition.

Definition 9 *The* family $\mathcal{F}(\mathrm{M}, *)$ of fuzzy neuroidal nets *over the set* M *of truth degrees and t-norm $*$ is an infinite sequence $(\mathcal{N}_1, \mathcal{N}_2, \ldots)$ of fuzzy neuroidal nets, with $\mathcal{N}_n = (\mathrm{G}_n, \mathrm{I}_n, o_n, \mathrm{W}_n, \mathrm{Q}_n, \mathrm{T}_n, \Delta_n, \mathrm{M}, \mu_n, *)$.*

Definition 10 *Let $\mathcal{F}(\mathrm{M}, *)$ be a family of fuzzy neuroidal nets. We say that family $\mathcal{F}(\mathrm{M}, *)$ is polynomially bounded if and only if there is a polynomial $p(n)$ such that for each $n \geq 0$ the description complexity of $\mathcal{N}_n \in \mathcal{F}(\mathrm{M}, *)$ (i.e. the space, measured in bits, needed to write down a full specification of $\mathcal{N}_n$) is bounded from above by $p(n)$.*

Definition 11 *The* fuzzy language $L_{\mathcal{F}(\mathrm{M},*)}$ accepted *by the family of fuzzy neuroidal nets $\mathcal{F}(\mathrm{M}, *)$ is the fuzzy set of ordered pairs*

$$L_{\mathcal{F}(\mathrm{M},*)} =$$
$$= \cup_{n \geq 0}\{(w, e(w))|\ |w| = n \text{ and } w \text{ is accepted by } \mathcal{N}_n \text{ with truth degree } e(w)\}$$

Note that in order to define the above language for each input size a special neuroidal net is used. The class of fuzzy languages accepted by families of fuzzy neuroidal nets of polynomial size will be denoted as POLY-FNN. For crisp neuroidal nets we denote the respective class as POLY-CNN.

Fuzzy neural nets are restricted kind of fuzzy neuroidal networks in which the neuroids can modify neither their weights nor their thresholds. The respective set of neuroidal states consists of only two states — of a firing and quiescent state. Moreover, the neurons are forced to fire if and only if the excitation reaches the threshold value. The computational behaviour of neural networks and fuzzy languages recognized by fuzzy neural nets are defined similarly as those of neuroidal ones.

5 Equivalence of fuzzy Turing machines with fuzzy neuroidal nets

Our next aim will be to show that the class POLY-FNN coincides with the class Fuzzy-PSPACE/poly, i.e. with the class of languages accepted by polynomially space bounded fuzzy Turing machines with a polynomially bounded advice function. We will do this by sketching a mutual simulation of fuzzy neuroidal nets by fuzzy Turing machines with advice.

Theorem 1 *Let L be a fuzzy language. Then the following assertions are equivalent:*

- $L \in$ *POLY-FNN*
- $L \in$ *Fuzzy-PSPACE/poly*

Proof: Let $L \in$ POLY-FNN. This means that there is a polynomially bounded family $\mathcal{F}(\mathrm{M}, *)$ such that $L = L(\mathcal{F}(\mathrm{M}, *)) = L(\mathcal{F})$. We will design a Fuzzy-NTM machine $\mathcal{A} = (S, T, I, \Delta, b, q_0, q_f, M, \mu, *)$ 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 fuzzy neuroidal net $\mathcal{N}_n \in \mathcal{F}(\mathrm{M}, *)$. 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 neuroidal net in the previous section) thanks to the fact that both fuzzy Turing machine and the fuzzy neuroidal nets work with the same set M of membership degrees and the same t–norm. Doing so, nondeterministic actions of $\mathcal{N}_n$ are simulated nondeterministically by choosing the instructions of $\mathcal{A}$ with the membership degree corresponding to that of individual neuroid updates. The compatibility of acceptance mechanism of both devices guarantees that an input accepted by $\mathcal{N}_n$ is also accepted by $\mathcal{A}$ with the same truth degree. Thus, $L \in$ Fuzzy–PSPACE/poly.

Now we prove the opposite inclusion. We will prove a slightly stronger result than needed — we will prove the reverse simulation of a Fuzzy-NTM by a family of neural (rather than neuroidal) nets. Let $L \in$ Fuzzy–PSPACE/poly, let $\mathcal{A} = (S, T, I, \Delta, b, q_0, q_f, M, \mu, *)$ be a single-tape Fuzzy-NTM with a polynomial advice, with a separate input tape. Let $\mathcal{A}$ accept L, let f be the respective polynomial advice function of size $d(n)$, let $\mathcal{A}$ be of polynomial space complexity $p(n)$. For each input of size n we will construct a specific neural net $\mathcal{N}_n$ that will accept exactly the words w of size n as $\mathcal{A}$ does, and it will accept them with the same truth degree as $\mathcal{A}$ does.

The size of the neural net will be bounded by $q(n) = O(\max\{d(n), p(n)\})$. 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 $\mathcal{A}$. Of course, care must be taken to simulate moves of $\mathcal{A}$ by update instructions within $\mathcal{N}_n$ with the same truth degrees in order to achieve the truth-degree compatibility of corresponding IDs at both devices.

The simulation is further complicated by the fact that the neural net 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 neural net 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 and that it will accept them with the same truth degree as $\mathcal{A}$ does. Thus, we can construct an infinite polynomially bounded family $\mathcal{F}$ such that $L = L(\mathcal{F})$. It follows that $L \in POLY - CNN \subseteq POLY - FNN$.

□

Corollary 1 *For the fuzzy languages, POLY-FNN= Fuzzy-PSPACE/poly. For the crisp ones, POLY-CNN=PSPACE/poly.*

Proof: The first equality is a direct consequence of the previous theorem. The second equality is a special case of the first one when crisp instances of the respective devices are considered. Note that this result has been known before (cf. [8]).

□

So far our approach differs substantially from earlier attempts to implement computations of fuzzy finite automata by neural nets. Namely. e.g. Omlin et al. [7] have designed analog neural nets simulating fuzzy finite automata in a way in which the acceptance degree becomes the result of the respective computation. Contrary to that, in our approach we rigorously see fuzzy computations as computations to which the (degree of) fuzziness is assigned by a mechanism that lies outside the given computational mechanism, much in the same way as the truth degrees are assigned to formulae within a fuzzy proposition calculus (cf. [2]). Nevertheless, the construction of fuzzy neuroidal nets from the proof of Theorem 1 simulating a fuzzy Turing machine can be modified so that the acceptance truth degree will become the result of simulation. We will sketch the respective construction only for the practically interesting case of deterministic fuzzy computations. Namely, in such a case the acceptance criterion from Definition 8 simplifies to a computation of the acceptance truth degree along a unique computation path. To achieve this we must extend the net by neuroids that will (deterministically) compute the truth degree of each ID reached by the net during its computation on a given input. This can be done under the assumption that the respective t-norm is a computable function and that the elements of M have a finite representation (e.g. they are rational numbers). In fact, the resulting construction reminds much the construction of a crisp Turing machine from [13] that simulates a fuzzy Turing machine and at the same time computes the acceptance truth degree. As a result we get the following corollary to Theorem 1.

Corollary 2 *Let $M_q \subset \mathbb{Q}$ be a finite set of rational numbers, let $*_Q$ be a computable t-norm, let $\mathcal{F} = (S, T, I, \Delta, b, q_0, q_f, M_Q, \mu, *_Q)$ be a deterministic fuzzy Turing machine. Then for each input w of size n there is a neuroidal net that simulates $\mathcal{F}$ on that input and produces the acceptance degree $e(w)$ of w if and only if w is accepted by $\mathcal{F}$ with truth degree $e(w)$.*

6 Conclusions

We defined a new class of discrete fuzzy neuroidal nets whose computational mechanism is compatible with that of fuzzy Turing machines that represent the most general fuzzy computational mechanism. We also showed that both kinds of devices are computationally equivalent and we characterized precisely their computational efficiency. Our results open the way for considering simulations of fuzzy Turing machines by analog neural nets.

Bibliography

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

[2] Hájek, P.: Metamathematics of Fuzzy Logic. Kluwer, 1998

[3] Hopcroft, J. E. — Ullman, J. D.: Introduction to Automata Theory, Languages, and Computation. Addison–Wesley Publishing Company, Reading, Mass., 1979, 417 p.

[4] 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

[5] Lee, E. T. — Zadeh, L.A.: Note on Fuzzy Languages. Information Science, Vol. 1, No. 4, pp. 421–434, 1969

[6] Matescu, A. — Salomaa, A. — Salomaa, K. — Yu, S.: Lexical Analysis with a Simple Finite–Fuzzy–Automaton Model. J. Universal Comp. Sci., Vol 1, No. 5, pp. 292–311, 1995

[7] Omlin, Ch.W. — Thornber, K.L. — Lee Giles, C.: Fuzzy Finite–State Automata Can Be Deterministically Encoded into Recurrent Neural Networks. IEEE Trans. on Fuzzy Systems, Vol. 6, No. 1, pp. 76–89, February 1998

[8] 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.

[9] Santos, E.: Fuzzy Algorithms. Information and Control, Vol. 17, pp. 326–339, 1970

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

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

[12] Wiedermann, J.: Fuzzy Computations Are More Powerful Than Crisp Ones. Technical Report No. 828, Institute of Computer Science AS ČR, 2000, ftp://ftp.cs.cas.cz/pub/reports/v828-00.ps

[13] Wiedermann, J.: Fuzzy Turing Machines Revised. Submitted to a journal, June 2001

[14] Zadeh, L. A.: Fuzzy Algorithms. Information and Control, Vol. 12, No. 2, pp. 94–102, 1968