Fuzzy Computations Are More Powerful than Crisp Ones

Wiedermann, Jiří
2000

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Fuzzy Computations Are More Powerful Than Crisp Ones

Jiří Wiedermann

Technical report No. 828

December 2000

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

Fuzzy Computations Are More Powerful Than Crisp Ones

Jiří Wiedermann[1]

Technical report No. 828
December 2000

Abstract

A model of a Turing machine with a fuzzy instruction set is considered. It is a generalization of a standard nondeterministic machine which accepts words of some (fuzzy) language with various truth degrees which depend on the choice of so–called t–norm. This norm defines the corresponding fuzzy propositional calculus which is shown to be related in a certain reasonable way to the fuzzy Turing machine at hand. Next it is shown that a variant of fuzzy Turing machines which corresponds to the so–called Gödel norm is computationally equivalent to a Turing machine with fuzzy final states. A surprising result showing that the nondeterministic variants of the latter machine also accept fuzzy languages that are not recursively enumerable (r.e.). This can be compared with the power of deterministic fuzzy Turing machines that accept only r.e. languages. The respective results are in a sharp contrast to crisp computations where there is no difference between the computational power of deterministic and nondeterministic computations. The languages accepted by deterministic fuzzy Turing machines are further characterized as r.e. languages for which there exists a computable, w.r.t. the respective t–norm, mapping that for each word in the language assigns its respective truth degree. The respective results shed new light on the power of fuzzy computing and point to the fact that on deterministic devices nondeterministic fuzzy computations that are not a priori known to be time bounded cannot be simulated.

Keywords

fuzzy computing; fuzzy Turing machine; fuzzy propositional calculus; recursively enumerable languages

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

1 Introduction

In this paper we will concentrate on the investigation of computational power of fuzzy computations. Unfortunately, the formal notion of fuzzy computations seems to be quite fuzzy indeed. Attempts to define it have been around since the dawn of the notion of fuzzy algorithms by Zadeh in late seventies (cf.[7]). In those days fuzzy variants of Turing machines, Markov algorithms, and finite automata (cf. [6],[7]) were been proposed and also fundamentals of fuzzy language theory were established [3]. Surprisingly, despite of the rich application potential of fuzzy computing (e.g. as embodied in fuzzy control systems) no sufficient attention has been paid to the investigation of recursive–theoretical limits of fuzzy computations.

In order to do so one needs an agreed-upon machine model whose computations will serve as a paradigmatic example of fuzzy computations. No doubts that any attempt to define such a model should start at the same level as in the classical computability (or complexity) theory, i.e. within the classical framework of the Turing machine theory. In doing so the above mentioned earlier approaches enhanced by the recent progress in developing the sound (meta)mathematical foundations o fuzzy logic (cf. [1]) should be taken into account.

The benefits from having a formal, generally accepted, Turing-machine-like model of fuzzy computations are obvious: such a model would enable a systematic study of the power and efficiency of fuzzy computing, allow its comparison with standard models of computing and, last but not least, enable a transfer of the known results between the domain of crisp and fuzzy computing.

The aim of this paper is to propose a candidate model for computability and complexity investigations of fuzzy computations and to bring the first results along the respective lines. Similarly as earlier approaches (cf. [6] or [7]) this model is based on the generalization of a classic notion of a (nondeterministic) Turing machine. Moreover, it is designed so as to be compatible with recent developments aiming at rigorous mathematical foundations of fuzzy logic. Thanks to this fact the model is more general than its forerunners and covers a full spectrum of specific t–norms some of which were treated separately in the early days of fuzzy computing. On the other hand, thanks to its similarity to classical models this model also enables its comparison with the former models as far as its computational power is concerned.

In Section 2 we will define the basic variant of a fuzzy Turing machine — viz. the one that makes use of a fuzzy instruction set. It accepts its inputs with various truth degrees which also depend on the choice of the underlying t–norm. This norm presents the link to the respective fuzzy predicate calculus. The (classical) notion of a fuzzy language accepted by a fuzzy Turing machine will be extended to cover our more general notion of a fuzzy Turing machine. However, due to our more general definition of a fuzzy Turing machine a proof that the respective definition of acceptance is sound is needed. A further rationale of our general approach is exemplified in Section 3 by pointing out to the relation between the truth degree of certain fuzzy propositional formulae and accepting computations of fuzzy Turing machines. In Section 4 a 'classic' variant of a fuzzy Turing machine (namely the one proposed in early works by Zadeh [7] or Santos [6]) based on the so–called min-max (or Gödel) norm is investigated.

It is shown that this model is computationally equivalent to the variant of a fuzzy Turing machine which makes use of crisp instructions in combination with fuzzy final states. Next, in Section 5 a surprising result is shown. It states that the languages accepted by nondeterministic variants of fuzzy Turing machines need not be recursively enumerable. This is in contrast to the deterministic variants which accept languages recursively enumerable. Finally, in Section 6 the class of fuzzy languages accepted by deterministic fuzzy Turing machines is further characterized with the help of the standard recursivity theory. In Section 7 the merit of the presented results is discussed.

For easier understanding of fuzzy logics background the acquaintance with e.g. the second chapter of monograph [1] is to be recommended.

2 Fuzzy Turing machines

Inspired by the earlier proposals of fuzzy Turing machines (especially those from [6] and [7]) a basic variant of a fuzzy Turing machine will be defined. It will be obtained from the standard nondeterministic Turing machine by fuzzyfication of its 'instruction set'. However, the machine design will still be general in the sense that the truth degrees associated with each accepting computation will be 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 (see Section 3). By choosing a particular t–norm we will obtain specific variants of fuzzy Turing machines, among them also the one defined originally by Santos [6].

Next we will proceed to the definition of a fuzzy Turing machine. Prior to giving the respective definition we will introduce the definition of a *t–norm* (cf. [1]).

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

1. t *is continuous;*
2. $*$ *is commutative and associative, i.e for all* $x, y, z \in [0,1]$ *we have* $x * y = y * x$ *and* $(x * y) * z = x * (y * z)$;
3. $*$ *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* $x * y_1 \leq x * y_2$;
4. *for all* $x \in [0,1]$ *we have* $1 * x = x$ *and* $0 * x = 0$.

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 with a fuzzy instruction set *(FIS-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*;*

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, where $m = |\Delta|$ is the number of moves in Δ;*

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 continuous 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 $Q_0 \vdash^{\alpha_0} Q_1 \vdash^{\alpha_1} Q_2 \ldots \vdash^{\alpha_k} Q_k$, with $\alpha_0, \ldots, \alpha_k \in M$ 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 the respective computations. 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 $d(Q_t)$ denote the truth degree of the proposition *"after t steps, starting from Q_0 the instantaneous description of $\mathcal{F}$ is Q_t"*. The respective *evaluation function* d is defined as

$$d(Q_t) = \begin{cases} 1, & t = 0 \\ d(Q_{t-1}) * \alpha, & t > 0 \quad \text{and } Q_{t-1} \vdash^{\alpha} Q_t \end{cases} \qquad (*)$$

Thus, the 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. [1]).

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). In such a case we say that along the computational path corresponding to the sequence $Q_0, Q_1, \ldots, Q_q$ input w is accepted with truth degree $d(Q_q)$.

A FIS-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 FIS-NTM. The input string w is* accepted with truth degree $e(w)$ *by $\mathcal{F}$ if and only if*

$$e(w) = \max_{Q} \{d(Q) | Q \textit{ is an accepting ID reachable from the initial ID } Q_0\}$$

Next we will show that the previous definition is sound — i.e. that the maximum $e(w)$ of truth degrees of accepting IDs over all accepting paths always exists.

Towards that end consider the commutative ordered semigroup $G = \langle [0,1], *, \leq \rangle$, where $M = \{\alpha_i | \alpha_i < \alpha_{i+1}, 1 \leq i \leq k-1\} \subset [0,1]$ is the set of instruction membership degrees from Definition 2. Let $G(M)$ be a subsemigroup of G generated by M. Syntactically, the elements of $G(M)$ are formed by 'products' of elements of M. Referring to the commutative and associative properties of the respective t-norm the elements of $G(M)$ take the form $\alpha_k^{s_k} * \alpha_{k-1}^{s_{k-1}} * \ldots * \alpha_1^{s_1}$, with $1 \geq \alpha_k > \alpha_{k-1} > \ldots > \alpha_1 > 0$ and $s_i \geq 0$ for $1 \leq i \leq k$. In the previous expression we wrote α^k to denote the product $\alpha * \alpha * \ldots * \alpha$ having k factors. Each element $\alpha \in G(M)$ of the previous form is uniquely determined by an integer k-tuple $\mathbf{s} = (s_k, s_{k-1}, \ldots, s_1)$. Such a k-tuple will be called a *tuple representation* (w.r.t. the subalgebra $G(M)$) of the respective element. The *length* of a tuple $\mathbf{s}$ is $len(\mathbf{s}) = s_k + s_{k-1} + \ldots s_1$. Let $\tau_k[\alpha_k, \alpha_{k-1}, \ldots, \alpha_1] : \mathbf{Z}^k \to [0,1]$ be the function that to each k-tuple representation of an element of $G(M)$ assigns the respective real number in [0,1]. Instead of $\tau_k[\alpha_k, \alpha_{k-1}, \ldots, \alpha_1]$ we will also write $\tau_k[M]$ or simply τ_k. Thus, under the previous notation, for element $\alpha \in G(M)$ represented by k-tuple $\mathbf{s}$ we have $\tau_k(s) = \alpha$.

Over the tuple representation of elements of $G(M)$ we will consider the partial order '$\sqsubseteq$' defined as follows: we will say that tuple $\mathbf{s} = (s_1, s_2, \ldots, s_k)$ *dominates* tuple $\mathbf{r} = (r_1, r_2, \ldots, r_k)$ (written as $\mathbf{r} \sqsubseteq \mathbf{s}$), or that $\mathbf{r}$ is *dominated by* $\mathbf{s}$ if and only if $r_i \leq s_i$

for $i = 1, 2, \ldots, k$. If an element dominates another one we say that the two elements are comparable. Considering the properties of the respective t-norm we can prove the following proposition.

Proposition 1 *If* $\mathbf{r} \sqsubseteq \mathbf{s}$ *then* $\tau_k(\mathbf{r}) \geq \tau_k(\mathbf{s})$.

A set D of k-tuples from $G(M)$ is called *independent* if and only if no two its elements are comparable.

Proposition 2 *For any* $k \geq 1$, *with* $|M| = k$, *any independent subset of* k*-tuples from* $G(M)$ *is finite.*

Sketch of the proof: Assume that there would be an infinite independent subset D_k of k-tuples of $G(M)$. We will show that then there must exist infinite independent sets of i-tuples for any $1 \leq i < k$.

Choose any $\mathbf{s} = (s_1, s_2, \ldots, s_k)$ from D_k and consider a decomposition of D_k into all subsets consisting of elements whose i-th component is fixed to some value between 0 and s_i, for $i = 1, 2, \ldots, k$. Each k-tuple $\mathbf{r} \in D_k$ must fall into at least one of such subsets since, thanks to independence of D_k, at least one of the components of $\mathbf{r}$ must be less than the corresponding component of $\mathbf{s}$. There is a finite number of such subsets (which need not be mutually disjoint) and their union equals D_k. Because there is at most a finite number of such subsets, some of them must be infinite, since otherwise D_k would be finite. Choose any such infinite subset corresponding to some fixed component value of its i-th component and remove this component from all the corresponding k-tuples. As a result we get an infinite independent set D_{k-1} of $(k-1)$-tuples.

Now repeat the previous construction with set D_{k-1} until we get the decomposition of the given infinite independent set into a finite number of independent subsets of a finite size. But this would contradict the assumption on the infiniteness of the original set. Note that at the latest such a situation will occur when we reach $k = 1$ since there is no infinite independent set of elements which consist of a single component.

Thus, the assumption that D_k was an infinite independent set was wrong.

□

Next we show that that $G(M)$ is well-ordered w.r.t. the standard ordering '$\leq$', i.e. each subset of $G(M)$ has a maximal element.

Lemma 1 *Let* F *be a subset of* $G(M)$. *Then* F *contains a maximal element, i.e. there is an element* $\mathbf{a} \in F$ *such that* $\mathbf{x} \leq \mathbf{a}$ *for all* $\mathbf{x} \in F$.

Sketch of the proof: Assume that there is no maximal element in F. In this case to each element $\mathbf{x_1} \in F$ there would be element $\mathbf{x_2} \in F$, with $\mathbf{x_1} < \mathbf{x_2}$, etc. Thus, there would be an infinite increasing chain $\mathbf{x_1} < \mathbf{x_2} < \ldots$ of elements of F. We show that in this chain there must exist i and j such that $i < j$ and $\mathbf{x_i} \geq \mathbf{x_j}$.

To see this consider the infinite sequence of k-tuples corresponding to the elements of the previous chain. Since there is but a finite number of tuples of each length, in this sequence there must be an infinite sub-sequence consisting of tuples of non-decreasing length. In this sub-sequence tuples are either incomparable or there exist

pairs of comparable tuples. In the former case the respective tuples will form an infinite independent set which, thanks to Proposition 2 cannot exist. In the latter case let there be two indices i, j with $i < j$ and two comparable tuples $\mathbf{r}$ and $\mathbf{s}$ in the subsequence such that $\tau_k(\mathbf{r}) = \mathbf{x_i}$ and $\tau_k(\mathbf{s}) = \mathbf{x_j}$. In this case since $len(\mathbf{r}) \leq len(\mathbf{s})$ the only possibility concerning the relation between $\mathbf{r}$ and $\mathbf{s}$ is $\mathbf{r} \sqsubseteq \mathbf{s}$ and consequently $\tau_k(\mathbf{r}) \geq \tau_k(\mathbf{s})$ (by Proposition 1). But this means that $\mathbf{x_i} \geq \mathbf{x_j}$ what is a contradiction with the assumption of the respective sub-chain ordering.

It follows that there are no infinite increasing chains in F and therefore it must contain the maximal element.

□

Corollary 1 *Let $\mathcal{F}$ be FIS-NTM, let w be any input, let $\mathcal{A} \neq \emptyset$ be the set of all accepting IDs of $\mathcal{F}$ on input w, let d be the evaluation function. Then the set $\{d(A)|A \in \mathcal{A}\}$ contains an element with a maximal truth degree.*

Sketch of the proof: Consider the computational tree T of $\mathcal{F}$ on input w. To each path in T of form $Q_0 \vdash^{\alpha_{i_0}} Q_1 \vdash^{\alpha_{i_1}} Q_2 \ldots \vdash^{\alpha_{i_r}} Q_{r+1}$, with $r \geq 0$ and $\alpha_{i_j} \in M$, an element $\alpha_{i_0} * \alpha_{i_1} * \ldots \alpha_{i_r} \in G(M)$ is assigned. Hence, to each accepting ID $A \in \mathcal{A}$ there is a corresponding element $\alpha_A \in G(M)$ whose value equals to the truth degree of A. Let $F = \{\alpha_A|A \in \mathcal{A}\}$ be the set of elements corresponding to all accepting IDs in T. Clearly, $F \subseteq G(M)$ and hence, according to Lemma 1 there exists its maximal element α. Obviously, the corresponding ID A from $\mathcal{A}$ will get the maximal truth degree $d(A) \in (0, 1]$.

□

Now we will proceed to the definition of fuzzy languages.

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

$$L_{\mathcal{F}} = \{(w, e(w))|\ w \text{ is accepted by } \mathcal{F} \text{ with truth degree } e(w) > 0\}$$

The *time complexity* $T(n)$ of the respective computation is the minimal number of moves needed to accept any input of length n.

Note that in case when $\mu(\delta) = 1$ for all $\delta \in \Delta$ the FIS-NTM equals the standard NTM as defined e.g. in [2]. Such a machine is also called a *crisp Turing machine*.

3 Fuzzy Turing Machines and Fuzzy Propositional Calculus

Next we will give the 'fuzzy logic' rationale for the above definition of acceptance by a fuzzy Turing machine. Namely, one could ask why we have chosen in the definition of the truth degree of acceptance the operator $*$ that corresponds to the truth function of fuzzy logic connective & of strong conjunction and not that of implication $\Rightarrow$, say.

In Section 5 we will show quite a surprising result concerning the power of the fuzzy Turing machines. Therefore it is of importance to be quite sure that the underlying notion of a fuzzy TM is 'well-defined', to avoid the objection that the non-standard

power of fuzzy Turing machines stems from their non-natural definition. This is why we will next give a formal substantiation that the notion of fuzzy TMs corresponds well to our expectations when the truth degrees of the computations of fuzzy TMs are compared to the evaluations of formulae in fuzzy propositional calculus $PC(*)$ [1]. We will say that a formula of $PC(*)$ with free variables is *satisfiable* with truth degree α if and only if there is such an assignment of truth values to the respective free variables which makes the formula hold with truth degree α.

We will show that for each fuzzy Turing machine $\mathcal{F}$ and its input w there is a propositional formula w_0 in the underlying fuzzy logic that is satisfied with a certain truth degree if and only if $\mathcal{F}$ accepts w with the same truth degree. The details are given in the following theorem.

Theorem 1 *Let $\mathcal{F}$ be a FIS-NTM of polynomial time complexity, let w be an input to $\mathcal{F}$. Then there is a polynomial-time bounded algorithm that from $\mathcal{F}$ and w constructs a propositional formula w_0 in the calculus $PC(*)$ such that w_0 is satisfiable with truth degree α if and only if $\mathcal{F}$ accepts w with truth degree α.*

Sketch of the proof: The respective proof mirrors the proofs showing the NP-completeness of satisfiability problem for standard, 'crisp' computations (cf. [2]). Therefore we will only sketch the main ideas pointing to places where the proof declines from the standard one in order to reflect the properties of the underlying fuzzy logic or fuzzy Turing machine.

Suppose that $\mathcal{F}$ has states $q_1, q_2, \ldots, q_s$ and tape symbols $X_1, X_2, \ldots, X_p$. Let $p(n)$ be a polynomial denoting the time complexity of $\mathcal{F}$. Thus if $\mathcal{F}$ accepts its input w then it does so within $p(n)$ moves, with $n = |w|$. Therefore in such a case there is at least one sequence of IDs $Q_0, Q_1, \ldots, Q_q$ such that Q_0 is initial ID, $Q_{i-1} \vdash^{\alpha} Q_i$ for $1 \leq i \leq q$, Q_q is accepting ID, $q \leq p(n)$, and no ID has more than $p(n)$ tape cells.

We will construct the fuzzy propositional formula w_0 that 'simulates' a sequence of IDs entered by $\mathcal{F}$. Each assignment of true or false (represented by 1 or 0) to the variables of w_0 represents at most one sequence of IDs of $\mathcal{F}$, possibly not a legal sequence. Formula w_0 will take on a truth value $\alpha > 0$ if and only if the assignment to the variables represents a sequence of IDs leading to acceptance of w by $\mathcal{F}$ with truth degree α.

The following variables are used as propositional variables in w_0 (we assume that they always will take on only crisp values).

- $C\langle i, j, t\rangle$ is true if and only if the ith cell on $\mathcal{F}$'s tape contains symbol X_j at time t.
- $S\langle k, t\rangle$ is true if and only if $\mathcal{F}$ is in state q_k at time t.
- $H\langle i, t\rangle$ is true if and only if at time t the tape head is scanning tape cell i.

In all the above expressions we assume $1 \leq i \leq p(n), 1 \leq j \leq p, 1 \leq k \leq s$ and $0 \leq t \leq p(n)$.

There are thus $O(p^2(n))$ propositional variables. Next we will construct w_0 as a conjunction of seven formulae $A_1, A_2, \ldots, A_7$ to assert that $Q_0, Q_1, \ldots, Q_q$ is an accepting sequence of IDs, where each Q_i is of length $p(n) = q$. Asserting that $Q_0, Q_1, \ldots, Q_q$ is an accepting sequence of IDs, then it is tantamount to asserting that the respective formulae are satisfiable if and only if the following conditions are met:

A_1: the tape head is scanning exactly one cell in each ID;

A_2: each ID has exactly one tape symbol in each tape cell;

A_3: each ID has exactly one state;

A_4: at most one tape cell, the one scanned by the tape head, is modified from one ID to the next;

A_5: the change of state, tape contents, and head location between successive IDs is compatible with the next-move relation of $\mathcal{F}$;

A_6: the first ID is an initial ID, and

A_7: the last ID is a final ID.

All the above formulae hold with truth degree 1 (are crisp), except of formula A_5 that will 'mirror' the next-move instructions of $\mathcal{F}$ that may have truth degrees less than 1. Therefore we will take a closer look at that formula. It asserts that each successive ID of $\mathcal{F}$ is obtained from the previous ID by one transition allowed by a next-move $\delta \in \Delta$ of $\mathcal{F}$.

Let machine $\mathcal{F}$ at time t be in state k reading the symbol j in the ith cell and let $\delta = (k, j, k', j', d)$ be a possible move, with truth degree α. Construct a subformula $A_5^{ijki'j'k't}$ that asserts the following. If at time t the ith cell contains symbol j and the tape head is scanning the ith cell and $\mathcal{F}$ is in state k, then at time $t+1$ the ith cell will be rewritten by j', the state entered will be k' and the head move will relocate the head to position i', with $d = i' - i$. Formally,

$$A_5 = \prod\nolimits_{t=0}^{*q(n)-1} \bigvee\nolimits_{ijki'j'k't} A_5^{ijki'j'k't}$$

where the operator Π^* denotes the "iterated strong conjunction" of subformulae following the operator and

$$A_5^{ijki'j'k't} =$$

$$= C\langle i, j, t\rangle \& H\langle i, t\rangle \& S\langle k, t\rangle \& C\langle i, j', t+1\rangle \& S\langle k', t+1\rangle \& H\langle i', t+1\rangle \Delta(k, j, k', j', i' - i)$$

If satisfied at some time t and some position i, i.e. when the next move is given by the predicate $\delta(k, j, k', j', d)$, with truth degree α, then by the rules of formulae evaluation in PC$(*)$ this truth degree is to be assigned to the above subformula, and only to this subformula. Namely, the remaining subformulae in A_5 corresponding for the same time t to different choices of the remaining variables are vacuously false due to their very definition. Thus in A_5 for each t and for a possible move exactly one

subformula will hold with truth degree α. This degree will be assigned as a truth degree to the respective subformula. Therefore, in accordance with the rules of evaluation, the truth degrees of formulae in the underlying fuzzy propositional calculus (cf. [1]) the whole formula A_5 will hold with the truth degree that corresponds to the 'star' product (i.e. the one corresponding to the operator $*$) of the evaluations of all subformulae. However, thanks to the properties of t–norms (cf. 1) only truth degrees that are less than 1 affect the respective evaluation. Therefore, if satisfied by a certain choice of moves, the formula A_5 will get the same truth degree as the corresponding sequence of moves of $\mathcal{F}$.

The details of construction of all other formulae can be found e.g. in [2]. Due to their very semantics and thanks to the fact that all variables involved are crisp, all these formulae are crisp.

The final formula w_0 we are after is the conjunction $A_1 \& A_2 \& \ldots \& A_7$. A more detailed analysis will show that this formula can be represented in space $O(p^3(n) \log n)$. From the construction of formula w_0 it is clear that given the accepting sequence of IDs $Q_0, Q_1, \ldots, Q_q$ leading to acceptance of w with truth degree α one can obviously find an assignment of 0s and 1s to the propositional variables $C\langle i, j, t\rangle$, $S\langle k, t, \rangle$ and $H\langle i, t\rangle$ that will satisfy w_0 with truth degree α and this will be the maximal possible truth degree among all possible assignments. Conversely, given an assignment of values to the variables of w_0 which maximizes the truth degree α of w_0 we can easily find an accepting sequence of IDs that led to accepting w with truth degree α. Thus w_0 holds with truth degree α if and only if the truth degree of (the proposition) "$\mathcal{F}$ *accepts* w" is α.

□

The above result opens the way to the investigation of complete problems for polynomially time bounded fuzzy computations. Namely, we have just shown that for this class of fuzzy computations the problem of finding an assignment to free variables of a fuzzy propositional formula which maximizes its truth degree is a complete problem. However, we will not follow this line in this paper.

4 Turing Machines with Fuzzy Final States

Let $G = \langle [0,1], *, \leq \rangle$ be a commutative ordered semigroup, let $M = \{\alpha_i | \alpha_i < \alpha_{i+1}, 1 \leq i \leq k-1\} \subset [0,1]$ be the set of instruction membership degrees from definition 2, and finally let $G(M)$ be a subsemigroup of G generated by M.

Next we will focus our attention on a specific class of fuzzy languages which is defined by fuzzy Turing machines with a specific t-norm. Namely, we will be interested in t-norms for which the respective subsemigroup $G(M)$ has a finite number of elements. This is e.g. the case of a popular *Gödel t-norm* defined as $x * y = \min\{x, y\}$.[2]This restriction has a serious impact on the class of languages recognized by the respective fuzzy Turing machines. In fact, all words from a fuzzy language accepted by such a fuzzy Turing machine are accepted with only finitely many truth degrees:

[2]Also Łukasiewicz t-norm shares the latter mentioned property, but we will be interested only in the case of Gödel's t-norm.

Theorem 2 *Let $\mathcal{F} = (S, T, I, \Delta, b, q_0, q_f, M, \mu, *)$ be a FIS-NTM with the Gödel t–norm, let $M = \{\alpha_1, \alpha_2, \ldots, \alpha_k\} \subset [0, 1]$ be the respective set of truth degrees of all moves of $\mathcal{F}$. Then the fuzzy language accepted by $\mathcal{F}$ is*

$$L_{\mathcal{F}} = \cup_{i=1}^{k} \{(w, \alpha_i) | w \text{ is accepted by } \mathcal{F} \text{ with truth degree } \alpha_i\}$$

Sketch of the proof: Note that in accordance with its definition in the Gödel t-norm the truth degree $d(Q_t)$ of reaching the ID Q_t from Q_0 is a non–increasing function (in t) which takes only the values from set M irrespectively of the length of the respective computational path. Thus, if a word is accepted then it is accepted with the truth degree equal to an element of M.

□

This fact opens the road for proving the equivalence of a FIS-NTM with the Gödel t–norm (note that this is the fuzzy Turing machine defined in [6]) with another variant of a fuzzy Turing machine — viz. the one with fuzzy final states:

Definition 5 *A* nondeterministic single–tape fuzzy Turing machine with fuzzy final states *(FFS-NTM) is an eight–tuple $\mathcal{G} = (S, T, I, \Delta, b, q_0, F, \nu)$ where:*

1. *S, T, I, Δ, b and q_0 are defined as in definition 2*
2. *$F \subseteq S$ is the finite set of* final, *or* accepting *states;*
3. *$\nu : F \to [0, 1]$ is a function that to each $q_f \in F$ assigns a truth degree $\nu(q_f)$ of its membership in F.*

Remark: Fuzzy final state Turing machines are defined similarly as the corresponding fuzzy final state finite automata in [4].

A FFS-NTM works as a (fuzzy) language acceptor as follows. The input string w is *accepted with truth degree* α if and only if

1. there is an accepting sequence of moves such that $\mathcal{F}$, started in the initial ID, after performing the above mentioned sequence of moves, eventually enters an accepting ID in a final state $q_f \in F$;
2. $\alpha = \max\{\nu(q_f)\}$, where maximum is taken over all accepting states in which the computation of $\mathcal{G}$ on input w might end.

The *language accepted* by $\mathcal{G}$ is a set of words accepted by $\mathcal{G}$ with a positive truth degree. Because set F of final states is finite, all words accepted by a FFS-NTM are accepted with only finitely many truth degrees. In view of Theorem 2 this already indicates the following statement.

Theorem 3 *Any FIS-NTM $\mathcal{F}$ with the Gödel t–norm is computationally equivalent to some FFS-NTM $\mathcal{G}$, and vice versa.*

Sketch of the proof: Let $\mathcal{F} = (S_1, T, I, \Delta_1, b, q_0, q_f, M, \mu, *)$ be a FIS-NTM with the Gödel t-norm, let $M = \{\alpha_1, \alpha_2, \ldots, \alpha_k\} \subset [0,1]$ be the respective set of all truth degrees of elements of Δ_1. Define the FFS-NTM $\mathcal{G} = (S_2, T, I, \Delta_2, b, q_0, F, \nu)$ as follows. For each $q \in S_1$ construct the set of k states $\{q_1, q_2, \ldots, q_k\} \in S_2$. The idea behind is that each $q_i \in S_2$ will represent the state $q \in S_1$ in which the value of α_i is stored. This idea is implemented with the help of the next-move relation Δ_2 which is designed so as to store the values of function $d(Q_t)$ defined in the previous section by the relation $(*)$ in the respective states.

Now the simulation of $\mathcal{F}$ by $\mathcal{G}$ is straightforward. At each time the tape contents of both machines is identical. The difference is that at each time machine $\mathcal{G}$ keeps and updates in its current state the value of the respective truth degree $d(Q_t)$ for the ID Q_t reached by $\mathcal{F}$ at that time. This is possible since there is but a finite number of different values of $d(Q_t)$. Function ν is designed so as to return the truth value stored in the respective final state.

For the reverse simulation of $\mathcal{G}$ by $\mathcal{F}$, make the following modification in the definition of $\mathcal{F}$. Define a new final state $q_f \notin S_2$ and add it to S_1. Turn all moves of $\mathcal{G}$ into the moves of $\mathcal{F}$ by defining their truth degree equal to 1. For each $f \in F$ such that $\nu(f) = \alpha$ add to Δ_1 a new move of form $(f, s, q_f, s, 0)$ with truth degree α, for any $s \in T$. It is clear that the resulting machine $\mathcal{F}$ will be a FIS-NTM accepting the same fuzzy language as the FFS-NTM $\mathcal{G}$ does.

□

5 Separation of fuzzy computations from the crisp ones

At first superficial sight one could infer from their definition that the FFS-NTMs, being restricted in their use of fuzziness, are 'almost equal' to standard nondeterministic 'crisp' Turing machines. The mere difference between the two classes is that in the former one one has to keep track in what final states the computation might end, and to select the one with the highest membership degree in set F. Surprisingly, in the general case this condition is not a computable one. Consequently, we will show that in general the FFS-NTM cannot be simulated by crisp machines.

Consider a slightly modified crisp version of a FIS-NTM as defined in 2. This modified machine differs from the standard model of a NTM by the fact that it is allowed to have several final states, collected into a finite set F.

Definition 6 *Let $\mathcal{C}$ be a crisp nondeterministic Turing machine, let F be the finite set of its final states. Then the problem of deciding whether for any $\mathcal{C}$, any $f \in F$ and any input w machine $\mathcal{C}$ will accept w in state f is called the REACHABILITY PROBLEM.*

Then it is not hard to see that for such machines the following theorem holds:

Theorem 4 *The REACHABILITY PROBLEM is undecidable.*

Sketch of the proof: Let $(\mathcal{C}, f, w)$ be an input to the REACHABILITY PROBLEM. From $\mathcal{C}$ we can construct machine $\mathcal{C}'$ which is a standard NTM (having just a single final state) as follows. To the next move relation of $\mathcal{C}$ we add further moves leading from all states $f \in F$ (F is the set of final states of $\mathcal{F}$) to a single new final state q_f, leaving the tape content intact and performing no moves. It is clear that $\mathcal{C}$ accepts w in state $f \in F$ if and only if $\mathcal{C}'$ accepts w. Thus, we have reduced the REACHABILITY PROBLEM to the HALTING PROBLEM which is known to be undecidable (cf. [2]). □

Let $\mathcal{C}$ be a crisp NTM with set F of final states, let $\langle\mathcal{C}\rangle$ be its encoding (cf. [2] for the details of the encoding). Define now the language corresponding to the REACHABILITY PROBLEM as follows:

$$L_{reach} = \{(\langle\mathcal{C}\rangle, w, f) |\ \mathcal{C} \text{ accepts } w \text{ in state} f \in F\}$$

The following is almost obvious:

Corollary 2 *L_{reach} is a r.e. language that is not recursive.*

Next we will show that there is no 'FFS-NTM simulator', that is, there is no crisp (deterministic or non-deterministic) algorithm that would simulate an arbitrary FFS-NTM.

Theorem 5 *There is no crisp Turing machine that could simulate any given FFS-NTM.*

Sketch of the proof: Assume that there is crisp Turing machine R that, given the description $\langle\mathcal{G}\rangle$ of a FFS-NTM $\mathcal{G} = (S, T, I, \Delta, b, q_0, F, \nu)$, would simulate its action on any input w. That is, R would accept input w in a state corresponding in one-to-one manner to some state $f \in F$ if and only if w is accepted by $\mathcal{G}$ in state f (if w is not accepted by $\mathcal{G}$ R need not halt). We will show that then we could construct a machine Q recognizing L_{reach}. Thus, Q will stop for each w and will accept (reject) w if $w \in L_{reach}$ ($w \notin L_{reach}$).

There is one 'technical' problem related to the description of $\mathcal{G}$. Namely, the function values of function ν might be real numbers that, in general, have no finite description and therefore the description of $\mathcal{G}$ could be of infinite length. However, from the definition of acceptance of a FFS-NTM it is obvious that only the ordering of the respective numbers is of importance. Thus, w.l.o.g. we can assume that for the purposes of simulation encoding $\langle\mathcal{G}\rangle$ is of a special form. Instead of the description of the range of ν the set of final states is described in this order: $f_1, f_2, \ldots, f_k$ for some $k > 0$ and $0 < \nu(f_1) < \nu(f_2) < \ldots < \nu(f_k)$.

Let v be a word. If it has a syntax that does not correspond to words from L_{reach}, v is rejected by Q. Otherwise, if v is of form $v = (\langle\mathcal{C}\rangle, w, f)$ machine Q will work as follows.

Q first modifies (the 'program' of) machine $\mathcal{G}$ in the following way. It adds a new final state f_0 to the set F, with $0 < \nu(f_0) < \nu(f_1)$ and adds a new transition into f_0 right from the initial state q_0. Let $\mathcal{G}'$ be the machine obtained in this way. Clearly, the

new machine accepts each word, i.e. it halts on each input. The following is obvious: a word w is accepted by $\mathcal{G}$ in state f_i if and only if w is accepted by $\mathcal{G}'$ in state f_i, for some $i > 0$. The only difference makes the state f_0 : word w is not accepted by $\mathcal{G}$ if and only if w is accepted by $\mathcal{G}'$ in state f_0.

Q now invokes machine R to simulate $\mathcal{G}'$ on input w and 'sees' in what state $\mathcal{G}'$ accepts w. If w is accepted in state f then Q accepts w. Otherwise, if w is accepted in a state different from f Q will reject w. Note that the introduction of state f_0 was necessary in order to assume that the simulation of $\mathcal{G}'$ by R will always halt.

It is clear that Q recognizes L_{reach} and hence L_{reach} is a recursive language. Nonetheless, this is in contradiction with Corollary 2. Therefore the assumption of existence of a universal simulating machine R was wrong.

□

Corollary 3 *There exist non r.e. languages accepted by FFS-NTMs.*

Sketch of the proof: If the languages accepted by FFS-NTMs were r.e. languages, then one could 'simulate' any FFS-NTM on a given input in the following (not very efficient, indeed) way. Let $\mathcal{G}$ be a FFS-NTM which is to be simulated on input w, let L be the respective language. Then it is enough to recursively enumerate L until w is generated what will happen if and only if $w \in L$. Otherwise the 'simulating machine' will run forever and not accept its input.

□

Note that the assumption that machine $\mathcal{G}$ was a nondeterministic Turing machine was a crucial one in the previous theorem. Namely, the deterministic Turing machine with fuzzy final states can be simulated by crisp machines:

Theorem 6 *A FFS-DTM can be simulated by a crisp Turing machine.*

Sketch of the proof: Let $\mathcal{D}$ be a FFS-DTM given by its description under the same assumptions on ordering of its final states as in the previous theorem. Then the simulating deterministic machine Q has no problems in simulating $\mathcal{D}$ since if a word is accepted by $\mathcal{D}$ there is exactly one computational path leading to acceptance in some final state. If on an input $\mathcal{D}$ runs forever then Q will do the same.

□

Corollary 4 *Languages accepted by FFS-DTMs are r.e. languages.*

Corollary 5 *Nondeterministic fuzzy final state machines are more powerful than the crisp ones.*

Thus fuzzy nondeterminism is more powerful than fuzzy determinism, and also fuzzy nondeterminism is more powerful than the 'pure', crisp nondeterminism. This comes as a surprise. Note, however, that Matescu et al. [4] came to the same conclusion for the case of finite automata with fuzzy instructions set: their nondeterministic versions are more powerful than the deterministic ones. Surprisingly, this is not the case with fuzzy

final state finite automata where both variants — deterministic and nondeterministic ones — recognize the same class of languages.

From the computational point of view it is not clear whether 'our' result should be expected from a machine model to correspond to fuzzy logic. Namely, by this result the fuzzy nondeterministic Turing machines are 'too powerful' — they recognize languages beyond those recognized by standard Turing machines. Intuitively, this is so since the nondeterministic fuzzy Turing machines 'implement' the operation of finding a maximum of a set of real numbers, whose membership in that set is undecidable. However, with regard to Theorem 1 any objections against the machine model underlying this result are at the same time objections against the fuzzy propositional calculus underlying the model.

Finally, note that time–bounded fuzzy nondeterministic computations can be simulated by deterministic machines, thanks to the fact that there is a known upper bound (which is equal to the time bound) limiting the depth of the underlying nondeterministic computational tree that is to be traversed in looking for the accepting state.

6 Recursively Enumerable Fuzzy Languages

Next we will take a closer look at fuzzy languages accepted by fuzzy Turing machines as defined in Definition 4. We will be especially interested in answering the following question: what conditions should a language $L \subset \Sigma^*$ and the respective truth function $e : \Sigma^* \to [0,1]$ satisfy in order to generate a fuzzy language $L_{fuzzy} = \{(w, e(w))|w \in L, e(w) > 0\}$ that is an r.e. language (in the standard sense). Such a fuzzy language will be called a *recursively enumerable fuzzy language*.

Thanks to Corollary 3 we know that in general a language accepted by a FIS-NTM need not be a r.e. fuzzy language. Thus we will concentrate to languages accepted by FIS-DTMs.

To answer the above posed question we will investigate the nature of $e(w)$. For that purpose we will similarly as in Section 2 consider the commutative ordered semigroup $G = \langle [0,1], *, \leq \ \rangle$ and its ordered subsemigroup $G(M)$. It is clear that the truth degree $e(w)$ of any word w accepted by a fuzzy Turing machine is an element of the subsemigroup $G(M)$. This fact will enable the characterization of fuzzy languages accepted by some FIS-DTM in terms of standard computability notions.

There is a technical problems related to computation of elements of $G(M)$ which in general are reals and therefore cannot be manipulated and represented straightforwardly by a Turing machine. However, in Section 2 we have already introduced the tuple representation of elements of $G(M)$. Thus, within $G(M)$, instead of computing directly with real numbers a Turing machine can compute with the help of the respective tuple representation.

Then the following theorem holds (function τ_k used in the statement of the theorem has been defined in Section 2, in the prelude to Proposition 1).

Theorem 7 *Let $e : T^* \to [0,1]$ be a truth function. Fuzzy language*

$$L_{fuzzy} = \{(w, e(w))|w \in T^*, e(w) > 0\}$$

is a r.e. fuzzy language if and only if

1. *the language* $L = \{w | \exists x \in [0,1] : (w,x) \in L_{fuzzy}\}$ *is a r.e. language and*

2. *there exists a set* M *of* k *real constants* $\alpha_1, \alpha_2, \ldots \alpha_k \in (0,1]$ *and a Turing machine which computes a function* $f : T^* \to \mathbf{Z}^k$ *that for each* $w \in L$ *assigns a tuple representation* $f(w)$ *of some element from* $G(M)$ *such that* $e(w) = \tau_k[M](f(w))$.

Sketch of the proof: Let L_{fuzzy} be accepted by some FIS-DTM. We will construct two crisp Turing machines $\mathcal{N}_1$ and $\mathcal{N}_2$, respectively, such that the first one will accept language L and the second one will compute ('define') function f as defined above. The inputs to $\mathcal{N}_1$ are arbitrary strings over T^* whereas the inputs to $\mathcal{N}_2$ are words from L.

Let $\mathcal{F}$ be a FIS-DTM machine that accepts the words of L_{fuzzy}. In particular, this machine defines the values of the set M. Machine $\mathcal{N}_1$ is simply the machine $\mathcal{F}$ with truth degrees of all its instructions set to 1. Clearly, $\mathcal{N}_1$ is a crisp deterministic machine that accepts L.

Machine $\mathcal{N}_2$ will be a deterministic transducer constructed as follows.

On input $w \in L$ machine $\mathcal{N}_2$ will follow the respective computational path of $\mathcal{F}$. Moreover, in addition to performing the same instructions it also will 'symbolically' compute the truth degrees of each ID reached by $\mathcal{F}$ (cf. the respective 'rule' $(*)$ for evaluation the truth degrees of IDs). Of course, this 'symbolic' representation of truth degrees of IDs is nothing else than a tuple representation of the respective element from $G(M)$. When reaching an accepting state of $\mathcal{N}_2$ we will define the value of $f(w)$ to be equal to the truth degree (in tuple notation) of the respective accepting ID in $\mathcal{F}$. Since $\mathcal{F}$ was a deterministic machine the final state at hand was the only one that could have been reached by $\mathcal{N}_2$ on input w. Therefore it holds that $\tau_k[M](f(w)) = e(w)$.

For the proof of the opposite implication assume that $\mathcal{N}_1$ is a Turing machine that accepts L. Let $\mathcal{N}_2$ be a transducer which to each $w \in L$ assigns a tuple representation $f(w) \in \mathbf{Z}^k$ of some element in $G(M)$. Then we will construct the FIS-DTM $\mathcal{F}$ that accepts the fuzzy language

$$L_{fuzzy} = \{(w, e(w)) | w \in L \text{ and } e(w) = \tau_k[M](f(w))\}$$

On input w machine $\mathcal{F}$ works as follows. First, it simulates $\mathcal{N}_1$ until it eventually reaches its accepting state. All the respective instructions are crisp. Then it switches to simulation of $\mathcal{N}_2$ on input w and computes a tuple representation $f(w)$, still making use of crisp instructions. It will end in some ID Q with truth degree 1. This ID corresponds to the final state of $\mathcal{N}_2$. Then $\mathcal{F}$ switches to 'fuzzy computation mode' and according to the tuple representation $f(w)$ it performs a series of transitions that end in an accepting ID Q_f with truth degree $\tau_k[M](f(w))$. The respective details are as follows. Let $(\ell_1, \ell_2, \ldots, \ell_k)$ be the respective tuple representation of $f(w)$. Call a transition a *dummy transition* if and only if it merely changes the state without changing anything else in a given ID. Then everything that $\mathcal{F}$ has to do, being in an ID Q, is to perform a series of transitions consisting of ℓ_i dummy transitions with truth degree α_i, for $i = 1, 2, \ldots, k$. It is clear that the computation will end in a unique ID Q_f (of $\mathcal{F}$) with truth degree $e(w) = \tau_k[M](f(w))$. Therefore $\mathcal{F}$ accepts L_{fuzzy}.

□

7 Conclusion

We have proposed a fuzzy variant of a nondeterministic Turing machine to serve as a formal model of fuzzy computations. This model is intended to be used in recursion and complexity-theoretic investigations aiming at the power, limits and efficiency of fuzzy computations, rather than designing efficient fuzzy algorithms. Our model differs from similar earlier proposals by its greater generality (it 'works' with any t-norm). Thanks to this the natural relationship between the fuzzy computations and fuzzy propositional calculus has been proved. This fact supports the evidence that our model is 'well defined', similarly as its previously defined special cases. We have also shown that a fully fledged fuzzy Turing machine, that is, its non-deterministic version, is computationally more powerful than its deterministic version. This is something that has no analogue in the standard realm of the recursion theory. Finally, we have shown a necessary and sufficient condition, in classical terms of computability theory, when a fuzzy language is accepted by some fuzzy Turing machine.

Our results indicate that fuzziness, seen as a computational resource does lead to increase in computational power when compared with crisp computations. If fuzzy logic underlies somehow the human thinking then one may speculate whether our results are related to the 'inexplicable' ability of human mind to solve 'unsolvable' problems (cf. [5]).

Nevertheless, returning to more prosaic matters, the building of the complexity theory of fuzzy computing seems to be the nearest future agenda for the computer science in the field of fuzzy computing. The present paper offers a basis for such investigations.

Acknowledgement.
The author is grateful to Petr Hájek for discussions related to the subject of the present paper and to his comments on earlier versions of this paper.

Bibliography

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

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

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

[4] 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

[5] Penrose, R.: The Emperors New Mind. Concerning Computers, Minds and The Laws of Physics. Oxford University Press, new York, 1989

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

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