CUTS, SLICES AND COMPUTABILITY.

R. Morales Bueno, J.L. Pérez de la Cruz, F. Triguero, R. Conejo
Dept. Lenguajes y Ciencias de la Computación. E.T.S.I. Informática.
Universidad de Málaga. Campus Teatinos 29071. Málaga. España.
e-mail: perez@lcc.uma.es

Abstract

In this paper several classes of fuzzy functions are studied and related to the computations of a fuzzy Turing machine. These classes are put into relation with the ones previously proposed in the literature.

Key words: fuzzy function, recursive function, computability, Turing machine.

1. INTRODUCTION

The computability of fuzzy functions has been the object of some attention since the first paper by E. Santos [8]; we must mention especially the work of Gerla [1,2]. Two main approaches have been followed to the concept: first, there are definitions of fuzzy computing devices or programs [8,9] [6,7]; on the other hand, there are definitions in terms of the recursiveness of the associated crisp functions [2,3]. Of course, both approaches converge.

In the present paper, we summarize work previously done and provide a new framework where these concepts fit. First, we prove that there are only three classes of functions definable in terms of recursiveness of cuts and slices (section 2) and also consider the bounded case (ibid.) These are new results. Then, in section 3 we will summarize definitions concerning fuzzy Turing machines and fuzzy computations. Some of these concepts have been presented in [5]. Finally, in section 4 the correspondence between both approaches is proven.

2. FROM α-CUTS TO RECURSIVENESS

In the following $W = \{0, \alpha_1, ..., \alpha_n\}$, $0 < \alpha_1 < ...< \alpha_n = 1$, will be an ordered finite semiring whose operations are $\oplus$ and $\otimes$ and whose order is $\leq$. The top of W will be the unit of $\otimes$, noted as 1. Anagolously, the bottom of W will be the zero of $\oplus$, noted as 0.

A fuzzy subset (W-set) of A is a (crisp) total function from A to W. We will note W-sets of A by greek lowercase letters subindexed by A. For example, μ_A, λ_A are W-sets of A. The α-cut of μ_A is the crisp set $\{x\in A \mid \mu_A(x)\geq\alpha\}$. The α–slice of μ_A is the crisp set $\{x\in A \mid \mu_A(x)=\alpha\}$.

A W-function from A to B is a W-set of $A\times B$. We will use greek lowercase letters subindexed by $A\times B$ to represent W-functions from A to B. The value $\phi_{A\times B}(x, y)$ can be interpreted as the degree with which the function $\phi_{A\times B}$ yields y when its argument is x. We will omit the subindex $A\times B$ when it is clear from the context. The image of $x\in A$ by a W-function $\phi_{A\times B}$ is the W-set of B $\zeta_{\phi,x}$ given by $\zeta_{\phi,x}(y) = \phi_{A\times B}(x, y)$.

We will only consider functions whose domain A and range B are effectively enumerable, i.e., there are computable bijections between N and A and between N and B. Therefore, A and B can be considered ordered by these bijections.

Definition 1. A W-function ϕ is partially recursive iff every α-cut ($\alpha\neq 0$) of ϕ is recursively enumerable.

Comment: Gerla [2] uses the denomination "partial recursive function". By restricting ϕ to its support, ϕ could be viewed as a partial function from $A\times B$ to W and hence this denomination makes sense. However, since ϕ is in fact total, we prefer the denomination "partially recursive function".

Definition 2. A W-function ϕ is pseudorecursive iff every α–slice ($\alpha\neq 0$) of ϕ is recursively enumerable.

Definition 3. A W-function ϕ is totally recursive iff every α-cut ($\alpha\neq 0$) of ϕ is recursive.

Comment: Gerla [2] uses the denomination "function with recursive graph". Our proposal aims to show the similarity with the remaining classes of recursive functions.

The proviso $\alpha\neq 0$ could be omitted in definitions 1 and 3. In fact, the 0-cut is the product $A\times B$, hence it is recursive. On the contrary, since the complementary of a recursively enumerable set may be not recursively enumerable, the restriction plays a substantial rule in definition 2.

Proposition 1. i) If ϕ is pseudorecursive, then ϕ is partially recursive; ii) if every α–slice of ϕ ($\alpha\neq 0$) is recursive, then ϕ is totally recursive.

Proof: i) the α_i-cut of ϕ ($\alpha \neq 0$) can be expressed by the union of the α_i-slice, α_{i+1}-slice, ... Therefore, if the α–slices ($\alpha \neq 0$) are recursively enumerable then the α-cuts ($\alpha \neq 0$) are recursively enumerable, too. The same argument -replacing "recursively enumerable" with "recursive"- proves ii). In fact, the finite union of recursively enumerable sets is recursively enumerable. The same assertion holds for recursive sets.

Proposition 2. If ϕ is totally recursive, then every α–slice of ϕ is recursive.

Proof: the α_i-slice of ϕ can be expressed by the difference between the α_i-cut and the α_{i+1}-cut. Therefore, if the α-cuts are recursive, then the α–slices are recursive, too.

Comment. These definitions and propositions can be summarized as follows: by combining some of the above conditions, namely that every α–slice/cut (including/excluding $\alpha=0$) is recursive/recursively enumerable, only three classes of *W*-functions arise: those of partially recursive, pseudorecursive and totally recursive functions. Moreover, if ϕ is totally recursive then ϕ is pseudorecursive and if ϕ is pseudorecursive then ϕ is partially recursive. Table 1 shows these results:

excluding $\alpha=0$	r. e.	α-cuts	partially rec.
excluding $\alpha=0$	r. e.	α–slices	pseudorec.
excluding $\alpha=0$	rec.	α-cuts	totally rec.
excluding $\alpha=0$	rec.	α–slices	totally rec.
including $\alpha=0$	r. e.	α-cuts	partially rec.
including $\alpha=0$	r. e.	α–slices	totally rec.
including $\alpha=0$	rec.	α-cuts	totally rec.
including $\alpha=0$	rec.	α–slices	totally rec.

Table 1.

Definition 4. A set $D \subseteq A \times B$ is recursively bounded iff

i) *D* is recursive.

ii) there exists a recursive function $f{:}A \to B$ such that for every $x \in A$ $\{(x, y) \in D\} \subseteq \{(x, y) \mid y \leq f(x)\}$.

Definition 5. A *W*-function ϕ is bounded recursive iff every α-cut ($\alpha \neq 0$) of ϕ is recursively bounded.

Comment: since the 0-cut of ϕ is $A \times B$ and hence infinite, the proviso $\alpha \neq 0$ cannot be omitted in this definition.

Proposition 3. If ϕ is bounded recursive, then every α–slice of ϕ ($\alpha \neq 0$) is recursively bounded.

Proof: the α_i-slice of ϕ can be expressed by the difference between the α_i-cut and the α_{i+1}-cut. Therefore, if the α-cuts ($\alpha \neq 0$) are recursive, then the α–slices ($\alpha \neq 0$) are recursive, too. On the other hand, let $f_1, ..., f_n$ be the bounding recursive functions of the $\alpha_1, ..., \alpha_n$ -cuts. Obviously, f_1 bounds every α–slice ($\alpha \neq 0$), hence every α–slice ($\alpha \neq 0$) is recursively bounded.

3. FUZZY COMPUTATIONS

In the following we summarize and arrange some concepts presented in previous papers [4] [5].

Definition 6.- A *W*-Turing machine (*W*-TM) is a 4-uple $Z=(V, S, p, s_0)$ where *V* and *S* are finite nonempty sets, $V \cap S = \varnothing$; $s_0 \in S$ and *p* is a *W*-function from $S \times V$ to $(V \cup \{L, R\}) \times (S \cup \{h\})$; $L,R \notin V$, $h \notin S$.

In the definition above *S* is the *state set* , s_0 the *initial state*, *V* the set of *simbols*. Moreover, *p* is the *transition W-function*, namely *p(s, c, z, s')* represents the degree of validity of the choice of the act (*z, s'*) at the state *s* and when the tape symbol *c* is scanned. Such an act is to move to the rigth (if *z* is *R*), to the left (if *z* is *L*) or to replace *c* by *z* (if $z \in V$) and successively to asume the state *s'* (stop if *s'* is *h*). We shall assume that in *V* there is a special symbol # that stands for blank.

For every $u \neq 0$, we can write $(s, c) \xrightarrow{u} (z, s')$ to denote that $p(s, c, z, s')=u$ and $(s,c) \to (z,s')$ to denote that $p(s, c, z, s')=1$. The *W*-function *p* is characterized by a finite set of arrows of the type above.

Definition 7. An *elementary description* is an element of $(\{\varepsilon\} \cup (V\text{-}\{\#\})V^*)\ (S \cup \{h\})V\ (V^*(V\text{-}\{\#\}) \cup \{\varepsilon\})$, i. e., a string of the form *usaw*, where *s* is a state (or the halting state *h*), *u* is the empty string or a string over *V* whose first symbol is not #, *a* is a symbol of *V*, and *w* is the empty string or a string over *V* whose last symbol is not #. An elementary description expresses the present state (*s*), the content of the tape (*uaw*) and the position and content of the scanned square (*a*). Elementary descriptions will be denoted by d, d_1, d_2, ... The set of all elementary descriptions of *T* will be denoted *D*(*T*). We will omit the argument *T* when no ambiguity can arise.

Definition 8. Let $d \in D$ be an elementary description and let *p* be the transition function of *T*. The set of successors of *d*, σ^d, is a *W*-set of *D* defined as follows:

i) if $d=uscw$, $d'=us'c'w$ then $\sigma^d(d')=p(s,c,c',s')$.

ii) if $d=uscc'w$, $d'=ucs'c'w$, then $\sigma^d(d')=p(s,c,R,s')$.

iii) if $d=usc$, $d'=ucs'\#$, then $\sigma^d(d')=p(s,c,R,s')$.

iv) if $d=uc'scw$, $d'=us'c'cw$, then $\sigma^d(d')=p(s,c,L,s')$.

v) if $d=scw$, $d'=s'\#cw$, then $\sigma^d(d')=p(s,c,L,s')$.
vi) otherwise $\sigma^d(d')=0$.

Definition 9. A description δ_D is a *W*-set of *D*. δ_D is an initial description iff $\delta_D=\{1/s_0\#\}$.

Definition 10.- An elementary description $d=usaw$ is *final* iff $s=h$. We will call *u* the output string of *d*. A description δ_D is final iff for every $d\in D$ such that $\delta_D(d)\neq 0$, *d* is an elementary final description.

Let us define now the transitions between fuzzy descriptions. Notice that since every elementary description yields a finite number of elementary descriptions in one step, a fuzzy description with finite support must yield another finite description.

Definition 11.- We say that *T* generates δ' from δ in one step ($\delta \Rightarrow_T \delta'$) iff δ is not a final description and δ' is the fuzzy subset of *D* given by $\delta'(d') = \sum_{d\in D} \delta(d) \otimes \sigma^d(d')$.

Definition 12.- Let *T* be a *W*-TM. The sequence generated by *T* with input *x* is $\{\delta_i(x)\}$ defined as follows: a) $\delta_0(x) = \{1/s_0\, x\}$; b) $\delta_i(x) \Rightarrow_T \delta_{i+1}(x)$.

For every description δ_i we can define a *W*-set μ which can be thought as already computed by *T*. Let us call it the "output set" of δ_i. Formally:

Definition 13.- Let δ be a description. Let $d=uhaw$ be a generic final elementary description. The output set of δ is the *W*-set of *V** μ given by $\mu(u)=\sum_{a,w}\delta(uhaw)$. If the sequence generated by *T* with input *x* is $\{\delta_i(x)\}$, then the output sequence of *T* with input *x* is $\{\mu_i(x)\}$ where every $\mu_i(x)$ is the output set of $\delta_i(x)$.

Notice that these sequences may be finite or infinite and always are fuzzy chains (nondecreasing w. r. t. fuzzy set inclusion). Since *W* is a finite lattice, the set of *W*-sets of *V** is a complete lattice and there is always a lub of $\{\mu_i(x)\}$ given by $\mu_X = \cup\, \mu_i(x)$.

Definition 14.- Let *T* be a *W*-TM and μ a *W*-set of V*. *T* computes μ with input *x* iff $\mu = \mu_X$. Let ϕ be a *W*-function from *U** to *V**. If for every $x\in U^*$, *T* computes the image $\zeta_{\phi,x}$ we say that *T* computes ϕ and ϕ is computable. Note: In [5] the same concepts were phrased as "*T* politropically computes ϕ and ϕ is politropically computable".

Definition 15.- Let *T* be a *W*-TM and μ a *W*-set of *V**. *T* computes monotropically μ with input *x* iff i) $\mu = \mu_X$; ii) for every step *i* and every element *y* of *V**, if $\mu_i(x)(y)\neq 0$ then $\mu_i(x)(y) = \mu_X(y)$. Part ii) could be stated as follows: for every element *y* of *V** the degree $\mu_X(y)$ is computed in a single step.

Let ϕ be a *W*-function from *U** to *V**. If for every $x\in U^*$, *T* computes monotropically the image $\zeta_{\phi,x}$ we say that *T* computes monotropically ϕ and ϕ is monotropically computable. Obviously, if *T* computes monotropically μ with input *x*, *T* computes μ with input *x*, and if ϕ is monotropically computable, then ϕ is computable.

Definition 16.- Let *T* be a *W*-TM and μ a *W*-set of *V**. *T* computes monotonically μ with input *x* iff i) $\mu = \mu_X$; ii) for every step *i* and every $y\in V^*$, if $\mu_i(x)(y)\neq 0$ then $\mu_i(x)(y') = \mu_X(y')$ for every $y'\in V^*$ such that $y'\leq y$; iii) if the support of μ_x is finite, then $\{\mu_i(x)\}$ is a finite sequence. Part ii) could be stated as follows: for every element *y* of *V** the degree $\mu_X(y)$ is computed in a single step, and at that step the degree of every $y'< y$ has been already computed. Note: In [5] the third condition was not stated, so affecting the correction of the proposition corresponding to the proposition 5 of present paper.

If for every $x\in U^*$, *T* computes monotonically the image $\zeta_{\phi,x}$ we say that *T* computes monotonically ϕ and ϕ is monotonically computable. Obviously, if *T* computes monotonically μ with input *x*, *T* computes monotropically μ with input *x*, and if ϕ is monotonically computable, then it is monotropically computable.

Definition 17.- Let *T* be a *W*-TM and μ a *W*-set of *V**. Let $\{\mu_i\}$ be the output sequence of *T* with input *x*. *T* computes finitely μ with input *x* iff $\{\mu_i(x)\}$ is finite and $\mu=\mu_X$.

Let ϕ be a *W*-function from *U** to *V**. If for every $x\in U^*$, *T* computes finetely the image $\zeta_{\phi,x}$ we say that *T* computes finetely ϕ and ϕ is finitely computable. Note: In [5] the same concepts were phrased as "*T* constructs ϕ and ϕ is constructible".

4. EQUIVALENCES BETWEEN RECURSIVENESS AND COMPUTABILITY.

Lemma 1. Let *T* be a crisp machine enumerating a sequence $S=\{(u_i, v_j)\}$ of elements of $U^*\times V^*$. Then for every $\alpha\in W$ there exists a *W*-TM T'_α such that $\mu_{u_i}(v_j) = \alpha$ if there exists a pair in *S* of the form (u_i, v_j), otherwise $\mu_{u_i}(v_j) = 0$.

Proof (sketch): The computation of T'_α could be described as follows: with input u_i, T'_α simulates the computation of *T*. Whenever an output (w_i, v_j) is generated, the first component w_i is compared with u_i. If they are equal, T'_α outputs v_j with degree α and goes on simulating *T*; otherwise it goes on without any output. Notice that this is a monotropical computation.

Proposition 4. The *W*-function ϕ is computable iff it is partially recursive.

Proof: vd. [2] and [5].

Proposition 5. The *W*-function ϕ is monotropically computable iff it is pseudorecursive.

Proof (sketch): Let *T* be the *W*-TM that computes ϕ. For every $\alpha \neq 0$, let us define a crisp TM T'_α such that T'_α enumerates the α-slice. T'_α sequentially writes the elements in U^* in some order; for each one, it simulates *n* steps of the computation of *T*. "Dovetailing" in the elements of U^* and the number of steps, *T* halts and outputs (u, v) iff *v* is generated with degree α. This is an effective proceduce to enumerate the α-slice. On the other hand, let us assume that every α-slice is r. e ($\alpha \neq 0$). Then there exist *n* crisp TM $T_1, ..., T_n$ that enumerate the α-slices. By lemma 1, there also exist *n* *W*-TM $T'_1, ..., T'_n$ whose parallel composition is able to compute motrotropically the whole output.

Proposition 6. The *W*-function ϕ is monotonically computable iff it is totally recursive.

Proof: vd. [5], proposition 5.

Proposition 7. The *W*-function ϕ is finitely computable iff it is bounded recursive.

Proof (sketch): In [4] it is proven that ϕ is finitely computable iff ϕ is totally recursive and there exists a crisp recursive function $s_\phi(x)$ such that for every *x*, $s_\phi(x) = sup(\zeta_{\phi,x})$. So, it suffices to show that if ϕ is bounded recursive, then $s_\phi(x)$ is recursive. If ϕ is bounded recursive, then the first α-cut $A_1 = \{(x, y) \mid \phi_{A\times B}(x, y) > 0\}$ is such that there exists a recursive function $f{:}A \rightarrow B$ such that $A_1 \subseteq \{(x, y) \mid y \leq f(x)\}$. From this *f* it is possible to compute in a finite number of steps all the possible images of *x*, hence to count the size of $\zeta_{\phi,x}$, so yielding a recursive $s_\phi(x)$.

Proposition 8. The *W*-function ϕ is computable iff it is *W*-computable (in Santos' sense).

Proof: By proposition 4, ϕ is computable iff it is partially recursive. In [2] it is proven that ϕ is computable (in Santos' sense) iff it is partially recursive.

Proposition 9. The *W*-function ϕ is monotropically computable iff it is strong partial recursive (in Gerla's sense).

Proof: By proposition 5, ϕ is monotropically computable iff it is pseudorecursive. Hence, there exists an effective procedure to set the value of $\phi(x, y)$, assuming that $\phi(x, y) \neq 0$ and $\phi(x, y)$ is computable restricted to $\phi(x, y) \neq 0$. Therefore it is *SPR*. In other words, the three concepts of monotropically computable, pseudorrecursive and strong partial recursive are the same.

5. CONCLUSIONS

An overview of some of the concepts of the theory of fuzzy computability has been presented and developed in this paper. Two threads, namely those of fuzzy Turing machines and recursiveness of cuts and slices, have been followed along the paper. The resulting, unifying view is somehow clear and concise. We think that little more could be done in the case of a finite set *W*.

Perhaps this work could be generalized for the case of a recursive set *W* provided with recursive order and operations. We are currently trying this extension.

REFERENCES

[1] L. Biacino and G. Gerla: Fuzzy subsets: A constructive approach. *Fuzzy Sets and Systems* **45** , 161-168 (1992).

[2] G. Gerla: Turing L-Machines and Recursive Computability for L-Maps. *Studia Logica* **XLVIII**, 179-192 (1989).

[3] L. Harkleroad: Fuzzy Recursion, Ret's and Isols. *Zeitschr. F.. Math. Logik Und Grundlagen D. Math.* **30**, 425-436 (1984).

[4] R. Morales-Bueno, J. L. Perez de la Cruz, B. Clares, and R. Conejo: Estudio de una clase de *W*-funciones calculables. *III Congreso Español Sobre Tecnologías y Lógica Fuzzy*, *Santiago de Compostela*, 1993, Santiago de Compostela, 1993, p. 153-160.

[5] Morales R., Pérez de la Cruz J. L., Clares B. , and Conejo R.: Acerca de la calculabilidad de las funciones difusas. *Actas Del IV Congreso De La Asociación Española De Tecnologías y Lógica Fuzzy*, *Blanes (Gerona)*, 1994, editores F. Esteva and P. García, Blanes (Gerona), 1994, p. 115-120.

[6] R. Morales, J. L. Pérez de la Cruz, R. Conejo, and B. Clares: A family of fuzzy programming languages. *Fuzzy Sets and Systems* **87**, 167-179 (1996).

[7] J. L. Pérez de la Cruz, R. Morales-Bueno, R. Conejo, and B. Clares: Caracterización mediante programas de ciertas clases de funciones *W*-calculables. *V Congreso Español Sobre Tecnologías y Lógica Fuzzy*, *Murcia*, 1995, Murcia, 1995, p. 141-146.

[8] E. S. Santos: Fuzzy algorithms. *Information and Control* **17**, 326-339 (1970).

[9] E. S. Santos, *Fuzzy Automata and Decision Processes*, editors M. M. Gupta, G. N. Saridis, and B. R. Gaines (North-Holland, New York, 1977), p. 133.