sciendo

Kairos. Journal of Philosophy & Science 20, 2018
Center for the Philosophy of Sciences of Lisbon University

A (Basis for a) Philosophy of a Theory of Fuzzy Computation

Apostolos Syropoulos
Xanthi, Greece
asyropoulos@gmail.com

Abstract Vagueness is a linguistic phenomenon as well as a property of physical objects. Fuzzy set theory is a mathematical model of vagueness that has been used to define vague models of computation. The prominent model of vague computation is the fuzzy Turing machine. This conceptual computing device gives an idea of what computing under vagueness means, nevertheless, it is not the most natural model. Based on the properties of this and other models of vague computing, an attempt is made to formulate a basis for a philosophy of a theory of fuzzy computation.

DOI 10.2478/kjps-2018-0009

1. Introduction

Following Raymond Turner [31], who starts his entry on the *Philosophy of Computer Science* in the Stanford Encyclopedia of Philosophy by quoting Stewart Shapiro [23, p. 525] in order to explain in a nutshell what is the philosophy of computer science, I will use the same excerpt from Shapiro's paper to give a general idea of what a philosophy of *fuzzy* computation should be. More specifically, given an arbitrary field of study X, Shapiro notes that

> "the main purpose of a given field of study is to contribute to knowledge, the philosophy of X is, at least in part, a branch of epistemology. Its purpose is to provide an account of the goals, methodology, and subject matter of X."

Before attempting to give an account of the goals and methodology of a theory of fuzzy computation, it is necessary to explain what is the subject matter of fuzzy computation. And before explaining what is fuzzy computation, it is absolutely necessary to explain why fuzzy computation does matter. In different words, why should we consider fuzzy computation as something important? Naturally, when it is made clear why fuzzy computing matters, then it would not be *easy* for anyone to argue that the theory of fuzzy computation should not be part of a general theory of computation. By admitting the omnipresence of vagueness, then it does make sense to argue that a theory of fuzzy computation *is* a general theory and, consequently, the ordinary theory of computation is a special case of this theory.

Unfortunately, the term *fuzzy computation* is considered to be a collective term that describes fuzzy arithmetic, fuzzy databases, fuzzy web searches, etc. However, these might be considered as applications of "vague-like" computation, but they do not belong to the body of a theory of fuzzy computation. For example, a fuzzy database operates in a non-vague environment and is supposed to handle vague data. Clearly, this is quite useful, but it is far from being a vague computation. In fact, to say that a fuzzy database is vague computing is like saying that a simulator of a quantum computer than runs on conventional hardware is able to achieve exactly what a real quantum computer can do[1]. A theory of fuzzy computation should propose tools to compute in a vague environment. These tools could be conceptual computing devices that operate in an environment, where, for instance, one cannot precisely measure the position of particles, and commands are executed to some degree. This last requirement is in accordance with the principles of fuzzy set the-

1 This is of course true once we accept the idea that quantum computers can achieve exactly what conventional hardware can achieve. However, this is not widely accepted (e.g., quantum computer can factorize large integers while for current conventional hardware it is difficult to perform this task).

ory. Obviously, we are not talking about a new computing paradigm but about a new approach to the essence of computation. Thus it would make sense to question the *limits of computation*. In order to study fuzzy computation we need fuzzy conceptual computing device. Although a fuzzy version of the Turing machine may seem like an ideal fuzzy conceptual computing device, I think it is not a natural model of vague computation. Nevertheless, people have used it to explore fuzzy computation.

Any new theory should not ignore predictions and results delivered by older theories. Instead, it should encompass them as limiting cases. For example, consider the general theory of relativity. This theory predicted things that were predicted by Newton's theory of gravity, but also it was able to make predictions about things not predictable by Newton's theory (e.g., gravitational waves) and explain things that Newton's theory could not precisely explain (e.g., the exact shape of Mercury's orbit). Thus, a theory of fuzzy computation should enrich current theories by incorporating vagueness into them.

Plan of the paper: Before providing an account of the goals and methodology of fuzzy computation, it is necessary to explain what is vagueness, in general, and fuzziness, in particular, and then to give an overview of fuzzy Turing machines.

2. What is Vagueness?

In the English language the word *fuzzy* is a synonym of the word *vague*. At least in colloquial usage, the term *vague* means something uncertain, imprecise or ambiguous. However, this is not what vagueness is about (see [33] for an exposition of vagueness). Formally, it is widely accepted that a term is vague to the extent that it has borderline cases, that is, cases in which it seems impossible either to apply or not to apply this term. The Sorites Paradox, which was introduced by Eubulides of Miletus, is a typical example of a puzzle that shows what it is meant by borderline cases. In addition, the paradox is one of the so-called *little-by-little arguments*. The term sorites derives from the Greek word for heap. The paradox is about the number of grains of wheat that makes a heap. All agree that a single grain of wheat does not comprise a heap. The same applies for two grains of wheat as they do not comprise a heap, etc. However,

there is a point where the number of grains becomes large enough to be called a heap, but there is no general agreement as to where this occurs.

Bertrand Russell [21] was perhaps the first thinker who gave a definition of vagueness:

> "*Per contra*, a representation is *vague* when the relation of the representing system to the represented system is not one-one, but one-many."

Based on this definition, one can say that a photograph that is so smudged, it might equally represent Brown or Jones or Robinson, is actually a vague photograph. Building on Russell's ideas, Max Black [4] had argued that most scientific theories, and of course any theory of computation cannot be excluded, are "ostensibly expressed in terms of objects never encountered in experience." For example, think that even the object oriented programming paradigm is about classes of object having fixed properties. In addition, it would be not an exaggeration to say that the Turing machine[2] is an idealization of some real-world system. However, there is no real-world system that operates like it and has the characteristics of the Turing machine. For example, we know that a universal machine accepts as input a string representing a number that denotes another machine and the input of the second machine. In the case of a machine that just adds two integers, one can verify that this number is:

$$G_n = 2^{249086222607801600000} \times 3^{6027271559398656000 \times 5^{1952835985245164544000}} \times 7^{1181345225642136576000} \times 11^{9568896327701306265600000}.$$

Let us compare this number with *the number of objects in the world* [32]:

(i) Planck-length = 10^{-35} m;

2 Roughly, the Turing machine is a conceptual computing device that consists of an infinite tape, which is divided into writable cells, a scanning head that can read the contents of a cell or print something to a cell, and the so-called *controlling device*, which is a lookup table that controls the behavior of the machine. Initially, one writes the input data into the cells of the tape, and then sets the machine into motion. The machine delivers a result if it stops after some finite amount of time. This simple machine is surprisingly powerful and it can be used to compute many functions and numbers. Not so surprisingly, the Turing machine is considered as the cornerstone of the (classical) theory of computation.

(ii) Planck-time = 10^{-43} s;

(iii) basic space-time region = 10^{-148} m^3s;

(iv) volume of the universe = 10^{78} m^3;

(v) age of the universe = 10^{18} s;

(vi) volume-age of the universe = 10^{96} m^3s; and finally

(vii) number of objects in the world = 10^{96} $m^3s/10^{-148}$ $m^3s = 10^{244}$.

Clearly, $G_n \gg 10^{244}$! Even if we can build a universal Turing machine, the scanning head could print or read symbols approximately because after a while ink would run out and symbols would be incorrectly printed on the tape. Of course we completely forget about power supply and other similar problems. The point here is that building a universal machine requires unrealistic resources. On the other hand, building a (computer) simulation of a Turing machine is something that can be easily done. However, the argument that modern computers are realizations of Turing machines is simply wrong because modern computers are interactive systems while Turing machines are not. Thus modern computers are systems that can simulate Turing machines while Turing machines cannot simulate them.

Black [4] proposed as a definition of vagueness the one given by Charles Sanders Peirce:

> "A proposition is vague when there are possible states of things concerning which it is intrinsically uncertain whether, had they been contemplated by the speaker, he would have regarded them as excluded or allowed by the proposition. By intrinsically uncertain we mean not uncertain in consequence of any ignorance of the interpreter, but because the speaker's habits of language were indeterminate."

I do not think that this definition is very useful because it assumes that vagueness is a linguistic phenomenon (see [12] for an overview of this narrow approach to vagueness). It is *obvious* that there are real life objects that exhibit some sort of vagueness. For instance, clouds are definitely vague objects as one cannot precisely specify their boundaries. In

addition, the "orbit" of an electron is a cloud with no precise boundaries. However, I will say more on vague objects on page 7. Readers interested in a general discussion of vagueness should consult a book like [7]. As far it regards the linguistic phenomenon, one could argue that although the term dirty is a linguistic term, still when it refers to cloths and the effort needed to clean them, the term ceases to be just a linguistic term and describes a real-world situation. Furthermore, washing machines that employ these "linguistic" terms to clean our cloths by making reason able use of energy, water, and detergent are not some imaginary things, but real appliances available to everyone. Of course, one could argue further that there is no vagueness at all and we can precisely describe dirt using some sort of scale. Indeed, but then we are talking about degrees of dirtiness, something that is chiefly modeled with fuzzy sets. For a more detailed discussion of these and other arguments against vagueness see [28].

It is widely accepted that there are (at least) three different expressions of vagueness [25]:

Many-valued Logics and Fuzzy Logic Borderline statements are assigned truth-values that are between absolute truth and absolute falsehood (see [24] for a book-length discussion of this idea).

Supervaluationism The idea that borderline statements lack a truth-value.

Contextualism The truth-value of a proposition depends on its context (i.e., a person may be tall relative to American men but short relative to NBA players).

There is a fourth, more recent, expression of vagueness that is based on the use of paraconsistent logics [7, 11]. In what follows, I will discuss only models of vague computation where vagueness is described using fuzzy set theory. The presentation of these models requires some familiarity with basic notion of fuzzy sets. The section that follows, briefly presents these ideas. Readers already familiar with fuzzy set theory can skip reading the next section.

3. Fuzzy Set Theory in a Nutshell

Fuzzy set theory was introduced by Lotfi Askar Zadeh [37] as an extension of ordinary set theory. Zadeh defined fuzzy sets by generalizing the membership relationship. In particular, given a universe X, he defined a fuzzy subset of X to be an object that is characterized by a function $A : X \to [0, 1]$. The value $A(x)$ specifies the degree to which the element x belongs to A. Let me now present the basic operations between fuzzy subsets.

Assume that $A, B : X \to [0, 1]$ are two fuzzy subsets of X. Then, their union and their intersection are defined as follows:

$$(A \cup B)\,(x) = \max[A(x), B(X)]$$

and

$$(A \cap B)\,(x) = \min[A(x), B(X)]$$

Also, if $\bar{A}$ is the complement of the fuzzy subset A, then $\bar{A}(x) = 1 - A(x)$. More generally, it is quite possible to use functions other than min and max to define the intersection and the union of fuzzy subsets. These functions are known in the literature as *t-norms* and *t-conorms*, respectively.

Definition 3.1 A t-norm is a binary operation $* : [0, 1] \times [0, 1] \to [0, 1]$ that satisfies at least the following conditions for all $a, b, c \in [0, 1]$:

Boundary condition $a * 1 = a$ and $a * 0 = 0$.

Monotonicity $b \leq c$ implies $a * b \leq a * c$.

Commutativity $a * b = b * a$.

Associativity $a * (b * c) = (a * b) * c$.

Definition 3.2 A t-conorm is a binary operation $\star : [0, 1] \times [0, 1] \to [0, 1]$ that satisfies at least the following conditions for all $a, b, c \in [0, 1]$:

Boundary condition $a \star 0 = a$ and $a \star 1 = 1$.

Monotonicity $b \leq c$ implies $a \star b \leq a \star c$.

Commutativity $a \star b = b \star a$.

Associativity $a \star (b \star c) = (a \star b) \star c$.

For more information on t-norms and t-conorms see [13] or any other textbook on fuzzy set theory.

4. The Subject Matter

Essentially, the classical theory of computation started with the publication of Alan Turing's *On Computable Numbers, with an application to the Entscheidungsproblem* [30]. Turing's paper introduced a conceptual computing device, that now bears his name, which was devised in order to solve the *Entscheidungsproblem*, that is, a problem that was put forth by David Hilbert in 1928. Roughly speaking, the *Entscheidungsproblem* asks if it is possible to find a method that will take as input a description of a formal language and a proposition in the language, the output of the method should be either *True* or *False* depending on the truth value of the proposition.

When introducing a new model of computation, it is almost customary to introduce some sort of Turing machine that accommodates the central idea behind the new model of computation. For example, in order to introduce probabilistic computing, Eugene Santos [22] introduced a probabilistic Turing machine. The same principle applies to quantum computing [10] and to non-deterministic computing (see [28] for a discussion of non-deterministic machines). However, not all Turing machine counterparts are very helpful. For instance, in quantum computing it is common to use quantum circuits in various theoretical studies because they can be used to directly express quantum algorithms.

The first steps towards a definition of a fuzzy Turing machine have been made by the inventor of fuzzy set theory [36]. Unfortunately, Zadeh provide "vague" definitions of fuzzy Turing machines and of fuzzy algorithms. One could say that he actually described what a fuzzy algorithm might be and, in a sense, speculated about the properties of a fuzzy Turing machine. However, his work prompted other researchers to investigate the notions of fuzzy Turing machines and fuzzy algorithms (see [28] for a comprehensive account of all these formulations). Although, fuzzy Turing machines are not the only model of fuzzy computation and, to some extend, not the most natural one, still they have been studied

thoroughly and thus it makes sense to give the definition of a fuzzy Turing machine.

The definition that follows was proposed Jiří Wiedermann [35] and I consider it the most complete and general definition of a fuzzy Turing machine:

Definition 4.1 A nondeterministic fuzzy Turing machine with a unidirectional tape is a nonuple

$$\mathcal{F} = (Q, T, I, \Delta, \text{␣}, \text{q}_0, \text{q}_f, \mu, *),$$

where:

- Q is a finite set of states;
- T is a finite set of tape symbols;
- I is a set of input symbols, where $I \subseteq T$;
- Δ is a transition relation and it is a subset of $Q \times T \times Q \times T \times \{L, N, R\}$. Each action that the machines takes is associated with an element $\delta \in \Delta$. In particular, for $\delta = (q_i, t_i, q_i{+}1, t_i{+}1, d)$ this means that when the machine is in state q_i and the current symbol that has been read is t_i, then the machine will enter state $q_i{+}1$, the symbol $t_i{+}1$ will be printed on the current cell and the scanning head will move according to the value of d, that is, if d is L, N, or R, then the head will move one cell to the left, will not move, or it will move one cell to the right, respectively.
- $\text{␣} \in T \setminus I$ is the blank symbol;
- q_0 and q_f are the initial and the final state, respectively;
- $\mu : \Delta \to [0, 1]$ is a fuzzy relation on Δ; and
- $*$ is a t-norm.

In order to fully understand how this conceptual machine works, it is necessary to present a few additional notions.

Definition 4.2 When μ is partial function from $Q \times T$ to $Q \times T \times \{L, N, R\}$ and T is a fuzzy subset of Q, then the resulting machine is called a deterministic fuzzy Turing machine.

A configuration gives the position of the scanning head, of what is printed on the tape, and the current state of the machine. If S_i and S_{i+1}

are two configurations, then $S_i \vdash^{\alpha} S_{i+1}$ means that S_{i+1} is reachable in one step from S_i with a plausibility degree that is equal to α if and only if there is a $\delta \in \Delta$ such that $\mu(\delta) = \alpha$ and by which the machine goes from S_i to S_{i+1}. When a machine starts with input some string *w*, the characters of the string are printed on the tape starting from the leftmost cell; the scanning head is placed atop the leftmost cell, and the machine enters state q_0. If

$$S_0 \vdash^{\alpha_0} S_1 \vdash^{\alpha_1} S_2 \vdash^{\alpha_2} \cdots \vdash^{\alpha_{n-1}} S_n,$$

then S_n is reachable from S_0 in *n* steps. Assume that S_n is reachable from S_0 in *n* steps, then the plausibility degree of this *computational path* is

$$D\big((S_0, S_1, \ldots, S_n)\big) = \begin{cases} 1, & n = 0 \\ D\big((S_0, S_1, \ldots, S_{n-1})\big) * \alpha_{n-1}, & n > 0 \end{cases}$$

Obviously, the value that is computed with this formula depends on the specific path that is chosen. Since the machine is nondeterministic, it is quite possible that some configuration S_n can be reached via different computational paths. Therefore, when a machine starts from S_0 and finishes at S_n in n steps, the plausibility degree of this computational path, which is called a *computation*, should be equal to the maximum of all possible computation paths:

$d(S_n) = \max[D((S_0, S_1, \ldots, S_n))]$.

In different words, the plausibility degree of the computation is equal to the plausibility degree of the computational path that is most likely to happen.

Assume that a machine starts from configuration S_0 with input the string *w*. Then, a computational path $S_0, S_1, \ldots, S_m$ is an accepting path of configurations if the state of S_m is q_f. In addition, the string *w* is accepted with degree equal to $d(S_m)$.

Definition 4.3 Assume that $\mathcal{F}$ is a fuzzy nondeterministic Turing machine. Then, an input string *w* is accepted with plausibility degree $e(w)$ by $\mathcal{F}$ if and only if:

· there is an accepting configuration from the initial configuration S_0 on input w;

· $e(w) = \max_S \{d(S) \mid S$ is an accepting configuration reachable from $S_0\}$.

Definition 4.4 The fuzzy language accepted by some machine $\mathcal{F}$ is the fuzzy set that is defined as follows:

$$L(\mathcal{F}) = \{(w, e(w))m|\text{w is accepted by } \mathcal{F} \text{ with plausibility degree } e(w)\}.$$

It is quite possible that a Turing machine might not write on its tape the symbol 1 but something that looks like it. Then, when reading this symbol the machine might have trouble deciding whether it is an 1 or not. Obviously, this a very interesting problem but it might happen because the read-write head is defective. If this happens randomly, then it should be necessary to include this "feature" in our analysis. However, this is an open problem at the moment.

Another model of fuzzy computation, more close to the idea of vagueness, are fuzzy P systems (see [28] for an up-to-date and thorough presentation of fuzzy P systems and other similar fuzzy models of computation). P system have been introduced by Gheorghe Păun [17]. Roughly, a P system is a model of computation that is based on the functionality of the cell. To the best of my knowledge, any cell has a membrane that surrounds it, separates its interior from its environment, regulates what goes in and out, etc. Inside the membrane, the cytoplasm takes up most of the cell volume. Various organelles (i.e., specialized subunits within a cell that have a specific function) are "floating" inside the cytoplasm. Just like a cell, a P system is enveloped in a porous membrane that allows objects to move in and out. Inside a P system there is an indefinite number of nested compartments, that is, compartments that may contain other compartments, etc., each of them enveloped by a porous membrane. Also, there is a designated compartment called the *output compartment*. In addition, each compartment may contain "solid," possibly repeated objects, that is, a multiset of objects, while it is associated with a set of multiset rewriting rules. The system operates in discrete time and these rules specify what changes can possible happen inside a compartment at each tick of the clock. In general, compartments cannot be deleted while objects may be multiplied, deleted, or introduced in a compartment. Computation stops when no rule is applicable and the result of the computation equals the number of objects that have been

accumulated in the output compartment. Now, a fuzzy P system is one where the multisets of objects are replaced by fuzzy multisets:

Definition 4.5 Assume that $M : X \rightarrow \mathbb{N}$ characterizes a multiset M. Then, a fuzzy multi-subset of M is a structure A that is characterized by a function $A : X \rightarrow \mathbb{N} \times [0, 1]$ such that if $M(x) = n$, then $A(x) = (n, i)$. In addition, the expression $A(x) = (n, i)$ denotes that the degree to which each of the n copies of x belong to A is i.

The cardinality of such a set is given by the following formula:

$$card\ A = \sum_{a \in A} A_m(a) A_\mu(a)$$

where $A_m(a) = n$ and $A_\mu(a) = i$.

Corollary 4.1 *The result of a computation delivered by a fuzzy P system is a positive real number.*

5. Goals and Methodology

Philosophical questions regarding the distinction between hardware and software and other similar questions are essentially the same when it comes to a philosophy of fuzzy computation. However, the important question here is: Where does vagueness come into play when one deals with real computers? In different words, is the hardware we use today non-exact or non-well-defined so that it is justified to see vagueness come into play?

It should not surprise anyone the fact that modern computers consume electricity that actually fluctuates. Typically, machines can cope with some small fluctuations of electricity because, among others, they have been designed to operate within a specific range of voltage. These fluctuations of electricity and other similar phenomena (e.g., noise in communication) are some sort of vague phenomena and, therefore, a raison d'être for vagueness. However, we intentionally choose to ignore this sort of vagueness and assume that our systems are exact when they are roughly exact. Of course this happens because those who built the first computers were not interested in vagueness and imprecision but rather in exactness and precision. Thus, one can safely conclude that

there is some sort of vagueness in hardware, which is left unexploited. The next big question is how can we make use of this vagueness? Or, going one step further, what would make a computer vague?

A computer that would harness vagueness in nature could be classified as a vague computer. But where do we really encounter vagueness in Nature? Surprisingly, based on remarks about the "fuzziness" of quantum mechanics (e.g., see [1, 19, 20]), one could argue that quantum computers are actually vague computers, provided one replaces probability theory with possibility theory or any other theory derived from fuzzy set theory (see [29] for details). But let us see how vagueness is manifested at the molecular, atomic or subatomic level. More specifically, E. J. Lowe [14] shows that vagueness exists in the subatomic level:

> "Suppose (to keep matters simple) that in an ionization chamber a free electron a is captured by a certain atom to form a negative ion which, a short time later, reverts to a neutral state by releasing an electron *b*. As I understand it, according to currently accepted quantum-mechanical principles there may simply be no objective fact of the matter as to whether or not *a* is identical with *b*. It should be emphasized that what is being proposed here is not merely that we may well have no way of telling whether or not *a* and *b* are identical, which would imply only an epistemic indeterminacy. It is well known that the sort of indeterminacy presupposed by orthodox interpretations of quantum theory is more than merely epistemic – it is ontic. The key feature of the example is that in such an interaction electron a and other electrons in the outer shell of the relevant atom enter an 'entangled' or 'superposed' state in which the number of electrons present is determinate but the identity of any one of them with a is not, thus rendering likewise indeterminate the identity of *a* with the released electron *b*."

The idea behind this example is that "identity statements represented by '$a = b$' are 'ontically' indeterminate in the quantum mechanical context" [38]. In different words, in the quantum mechanical context *a* is equal to *b* to some degree, which is one of the fundamental ideas behind

fuzzy set theory. Based on this observation, it should be clear that a vague computer should make use of vagueness as manifested in nature.

A vague computer should be able to run programs. Clearly, such a computing device should be able to run conventional programs, since, as expected, all vague models of computation would be far more general than their *crisp* counterparts. Forget for a moment the connection between vague and quantum computation and assume that there are no vague computers available at this moment. Then, does it make sense to talk about vague programs and vague programming languages today? The answer is emphatically *yes* and here is the rationale. Today, quantum computers are not widely available yet, still there are quantum-programming languages, like Quipper [9], which can be used to create quantum programs. Such programming languages are implemented using conventional methodologies and techniques (e.g., Quipper is written in Haskell) and programs expressed in these languages run on conventional hardware. Similarly, one can design and implement a vague programming language using a conventional programming language so that vague programs run on conventional hardware. However, one should note that since quantum computers are more powerful than conventional machines (e.g., they can compute true random numbers [18] whereas von Neumann machines can compute only pseudo-random number), it is of course impossible to use this extra power when working with such quantum programming languages. And of course this applies to vague programming languages too, provided that vague machines have similar capabilities.

Zadeh [36] has speculated about the form of commands in a vague programming language. Thus, according to Zadeh, a typical command of such a language would be "set y approximately equal to 10 if x is approximately equal to 5." Ever since a number of vague programming languages have been designed and implemented. RASP [8] was an extension of BASIC that provided basic operations for fuzzy sets but it did not include commands similar to the ones suggested by Zadeh's. The FLISP [26] programming language, an extension of LISP, provided facilities to input and process fuzzy data. For example, in order to enter the fuzzy set

$$Q = 0.3/2 + 0.9/3 + 1/4 + 0.8/5 + 0.5/6$$

where f/d means that $d \in_f Q$ (i.e, d belongs to Q to degree equals to f), one had to enter the following commands:

(SETQ U ‘ (0 1 2 3 4 5 6 7 8 9))

(FSETQ Q ((U) (FSET ((2@0.3) (6@0.5) (5@0.8) (3@–0.9) (4@1))))).

HALO [6] was a LISP-like language that employed many Pascal-like and C-like structures. In addition, the logical operations as well as some other operations are fuzzy. The assignment statement of the languages L and XL [15] allowed users to assign fuzzy numbers to variables. In addition, XL included a fuzzy repetition construct. Fril++ [3] is an object-oriented language where an object can be an instance of a class to some degree. This is a particularly interesting idea and implies that two instances of some class can be equal to some degree. And it would be quite interesting to see how one could implement this idea in a way similar to Java’s equals( ) method. Fuzzy Arden Syntax is programming language that has been designed “to provide an easy means of processing vague or uncertain data, which frequently appears in medicine” [34]. One can define fuzzy sets very easily:

U: = fuzzy set (2,0.3), (6,0.5), (5,0.8), (3,–0.9), (4,1).

The language allows commands that are reminiscent of Zadeh’s “commands”:

TempatureList: = read {temperature} where

it occurred within the past 24 hours

fuzzified by hours.

And of course, there is a fuzzy λ-calculus [2] where each term is associated with a degree and the b-reduction is redefined. I suppose the authors meant β-reduction, but that is just an educated guess… The notion of “approximately equal” can be introduced in a language by

means of an extended assignment operator (e.g., ~=) and an extended equality operator (e.g., ~==). Thus, a command like

x ~= y

would mean that x is actually assigned a compound value like y ± δy, where the δy should be implementation dependent or the user should be able to configure the compiler accordingly. In a sense, x would be an interval and not just a point.

The languages that were briefly reviewed can be roughly divided into two categories: those that allow the use of fuzzy sets and those that implement some basic principle of fuzzy set theory (e.g., similarity of objects). The languages that implement some sort of similarity with degree are closer to the spirit of vagueness. In general, fuzzy programming languages should provide facilities for the definition and manipulation of fuzzy sets. In addition, they should provide control structures that can handle fuzzy logical expression. Also, it is necessary to provide facilities to express similarities between structures.

In the previous section, I stated that Turing devised his (automatic) machine in order to solve the Entscheidungsproblem. This is a bit inaccurate. The truth is that Turing devised his machine and then he devised the universal Turing machine with which he gave his answer to the Entscheidungsproblem. In modern computer parlance, Turing essentially proved that it is not possible to tell whether a program that is not *responding* has entered a vicious circle or not. This problem is known as the *halting problem*. By showing that the halting problem is as hard as the Entscheidungsproblem, he proved that the Entscheidungsproblem is unsolvable, or better it is *Turing* unsolvable. Turing went one step further and made a bold statement—if a number or a function is computable, then it must be computable by a Turing machine. This statement is now known as the *Church-Turing thesis*.

Clearly fuzzy Turing machines form an extension of the classical archetypal conceptual computing device. Therefore, they should compute as many numbers and/or functions as their classical counterpart. The important question is whether these machines are more powerful than their classical counterparts. Jiří Wiedermann [35] has shown that fuzzy

Turing machines are more powerful than ordinary Turing machines. In different words, fuzzy Turing machines are conceptual computing devices that have *hypercomputational* capabilities [27]. Despite this *fact*[3] it is not clear at all what are the computational limits of these machines and how this will affect real world computing. After all, for each conceptual and logically consistent computing device there is a limit to what it can achieve. For example, Toby Ord and Tien D. Kieu [16] have argued that every logically consistent computing device cannot solve its own halting problem. Since there are no universal fuzzy Turing machines, it makes no sense to talk about their halting problem, nevertheless, it does make sense to try to find their computational limits.

Mark Changizi [5] assumes the validity of three theses or hypotheses: the Church-Turing thesis, the Programs-in-Head hypothesis, and the Any-Algorithm hypothesis. These hypotheses state:

Programs-in-Head Hypothesis For most natural language predicates P and their natural language negation '$\neg P$', their interpretations are determined by you using programs in the head.

Any-Algorithm Hypothesis You are free to choose from the set of all algorithms when interpreting natural language predicates or their natural language negations.

Changizi argues that these hypotheses together with the Church-Turing thesis imply the omnipresence of vagueness in language. Of course, it is not know if the Church-Turing thesis is valid and hypercomputation implies that it is not. Next, the two hypotheses are based on the assumption that mechanism is valid, which is equally problematic. In [27] I have argued against mechanism so I will not repeat these arguments here. What is even more problematic is that vagueness is restricted into language and because of Changizi's arguments any language is mostly vague. However, on a place with no intelligent beings there is no language and so no vagueness! Ergo, vagueness is not something real. And this is the reason I have tried to establish that vagueness is a fundamental property of the physical world and not some human invention.

3 See [28] for a discussion of various attacks to this result.

Conclusions

Fuzzy computing is a new branch of (theoretical) computer science that is not fully developed. There are a number of open questions regarding fuzzy conceptual computing devices and their capabilities. The answers to these questions greatly depend on one's philosophical prejudices. I have tried to briefly present the field of fuzzy computation and to discuss these open problems based on my own prejudices. In summary, there is no universal fuzzy Turing machine and it seems universality has nothing to do with vague computing devices. However, this should not pose an obstacle in the construction of vague computers, which should be based on vagueness as it appears at the particle level. Also, it seems that vague computing devices will be more powerful than their classical counterparts, but the upper bound of their computational power has not been determined yet.

References

[1] Granik, A. and Caulfield, H.J. (2001), Fuzziness in Quantum Mechanics, arXiv:quant-ph/0107054v1.

[2] Alvarez, D.S. and Skarmeta, A:F.G. (2004), A fuzzy language, *Fuzzy Sets and Systems*, 141, 335–390.

[3] James F. Baldwin, Trevor P. Martin, and Maria Vargas-Vera (1999). Fril++ a Language for Object-Oriented Programming with Uncertainty. In Anca L. Ralescu and James G. Shanahan (editors), *Fuzzy Logic in Artificial Intelligence*, volume 1566 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 62–78.

[4] Max Black (1937). Vagueness. An Exercise in Logical Analysis. *Philosophy of Science*, 4(4): 427–455.

[5] Mark A. Changizi (2003). *The Brain from 25,000 Feet*, volume 317 of Synthese Library. Springer Netherlands.

[6] David F. Clark (1991). HALO–a fuzzy programming language. *Fuzzy Sets and Systems*, 44:199–208.

[7] Richard Dietz and Sebastiano Moruzzi, editors (2010). *Cuts and Clouds: Vaguenesss, its Nature and its Logic*. Oxford University Press, Oxford, UK.

[8] Dragan D. Djakovic (1988). RASP–A Language with Operations on Fuzzy Set. *Computer Languages*, 13(3–4): 143–147.

[9] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron (2013). An introduction to quantum programming in quipper. In Gerhard W. Dueck and D. Michael Miller, editors, Reversible Computation, volume 7948 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 110–124.

[10] Mika Hirvensalo (2004). *Quantum Computing*. Springer-Verlag, Berlin, 2nd edition.

[11] Dominic Hyde and Mark Colyvan (2008). Paraconsistent Vagueness: Why Not? *The Australasian Journal of Logic*, 6:107–121.

[12] Rosanna Keefe and Peter Smith (editors) (1999). *Vagueness: A Reader*. The MIT Press.

[13] George J. Klir and Bo Yuan (1995). *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall (Sd).

[14] Edward Jonathan Lowe (1994). Vague Identity and Quantum Indeterminacy. *Analysis*, 54(2): 110–114.

[15] Rafael Morales-Bueno, José-Luis Pérez-de-la-Cruz, Ricardo Conejo, and Buenaventura Clares (1997). A family of fuzzy programming languages. *Fuzzy Sets and Systems*, 87: 167–179.

[16] Toby Ord and Tien D. Kieu (2005). The Diagonal Method and Hypercomputation. *The British Journal for the Philosophy of Science*, 56(1): 147–156.

[17] Gheorghe Păun (2002). *Membrane Computing: An Introduction*. Springer-Verlag, Berlin, Germany, EU.

[18] Pironio, S. and Acín, A. and Massar, S. and de la Giroday, A. Boyer and Matsukevich, D.N. and Maunz, P. and Olmschenk, S. and Hayes, D. and Luo, L. and Manning, T.A. and Monroe, C. (2010). Random numbers certified by Bell's theorem. *Nature*, 464: 1021–1024.

[19] Jarosław Pykacz (2011). Towards many-valued/fuzzy interpretation of quantum mechanics. *International Journal of General Systems*, 40(1): 11–21.

[20] Jarosław Pykacz, Bart D'Hooghe, and Roman R. Zapatrin. Quantum Computers as Fuzzy Computers (2001). In: B. Reusch, editor, *Fuzzy Days*

2001, volume 2206 of Lecture Notes in Computer Science, Springer, Berlin, Germany, EU, 526–535.

[21] Bertrand Russell (1923). Vagueness. *Australasian Journal of Philosophy*, 1(2): 84–92.

[22] Eugene S. Santos (1969). Probabilistic Turing Machines and Computability. *Proceedings of the American Mathematical Society*, 22(3): 704–710.

[23] Stewart Shapiro (1983). Mathematics and reality. *Philosophy of Science*, 50(4): 523–548.

[24] Nicholas J.J. Smith (2008). *Vagueness and Degrees of Truth*. Oxford University Press.

[25] Roy Sorensen. Vagueness. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2008 edition, 2008.

[26] Zenon A. Sosnowski (1990). FLISP–A language for processing fuzzy data. *Fuzzy Sets and Systems*, 37(1): 23–32.

[27] Apostolos Syropoulos (2008). *Hypercomputation: Computing Beyond the Church-Turing Barrier*. Springer New York, Inc., Secaucus, NJ, USA.

[28] Apostolos Syropoulos (2014). Theory of Fuzzy Computation. *IFSR International Series on Systems Science and Engineering*. Springer, New York.

[29] Apostolos Syropoulos (2017). On vague computers. In Andrew Adamatzky, editor, *Emergent Computation: A Festschrift for Selim G. Akl*, Springer International Publishing, Cham, Switzerland, 393–402.

[30] Alan M. Turing (1936). On Computable Numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42: 230–265.

[31] Raymond Turner. The philosophy of computer science. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2013 edition, 2013. http://plato.stanford.edu/archives/ fall2013/entries/computer-science/.

[32] Jean Paul Van Bendegem (1970). Why the largest number imaginable is still a finite number. *Logique et Analyse*.

[33] Kees van Deemter (2010). *Not Exactly: In Praise of Vagueness*. Oxford University Press.

[34] Thomas Vetterlein, Harald Mandl, and Klaus-Peter Adlassnig (2010). Fuzzy Arden Syntax: A fuzzy programming language for medicine. *Artificial Intelligence in Medicine*, 49: 1–10.

[35] Jiří Wiedermann (2004). Characterizing the super-Turing computing power and efficiency of classical fuzzy Turing machines. *Theoretical Computer Science*, 317: 61–69.

[36] Lotfi Askar Zadeh (1968). Fuzzy Algorithms. *Information and Control*, 12: 94–102.

[37] Lotfi Askar Zadeh (1995). Discussion: Probability Theory and Fuzzy Logic Are Complementary Rather Than Competitive. *Technometrics*, 37(3): 271–276.

[38] Steven French and Décio Krause (2003). Quantum Vagueness. *Erkenntnis*, 59: 97–124.