In Case of Interval (or More General) Uncertainty, No Algorithm Can Choose the Simplest Representative

Gerhard Heindl[1], Vladik Kreinovich[2], and Maria Rifqi[3]

[1]Bergische Universität GH Wuppertal
Fachbereich Mathematik
Gaussstr. 20
42097 Wuppertal, Germany
email heindl@math.uni-wuppertal.de

[2]Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
email vladik@cs.utep.edu

[3]LIP6, Pole IA
Université Pierre et Marie Curie, Case 169
4 place Jussieu
75252 Paris Cédex 05, France
email Maria.Rifqi@lip6.fr

Abstract

When we only know the interval of possible values of a certain quantity (or a more general set of possible values), it is desirable to characterize this interval by supplying the user with the "simplest" element from this interval, and by characterizing how different from this value we can get. For example, if, for some unknown physical quantity x, measurements result in the interval $[1.95, 2.1]$ of possible values, then, most probably, the physicist will publish this result as $y \approx 2$. Similarly, a natural representation of the measurement result $x \in [3.141592, 3.141593]$ is $x \approx \pi$.

In this paper, we show that the problem of choosing the simplest element from a given interval (or from a given set) is, in general, not algorithmically solvable.

1 In Case of Interval (or More General Set) Uncertainty, a User Would Like to Have a Representative Value from This Interval (Set)

The value of a physical quantity y is usually obtained either by a direct measurement, or by an *indirect measurement*, i.e., by processing the results of some related measurements $x_1, \ldots, x_n$. Since measurements are normally not 100% precise, their results may differ from the actual values of the measured quantities. As a result, after the measurement (direct or indirect), we do *not* get the *exact value* of the desired quantity, we only get a *set* Y of its possible values.

In many cases, this set Y is an interval, but more complicated sets are also possible: e.g., if we know that $y^2 = x_1$, and that $x_1 \in [1, 4]$, then the set of possible values of y is the union of two intervals $[-2, -1] \cup [1, 2]$.

For each set, it would be very convenient for the users of this information, if we could select a *representative* element of this set and describe the possible deviations from this value.

For example, if the set of possible values is the interval $[3, 5]$, it is natural to take the midpoint of this interval (i.e., the number 4) as the desired representative, and describe the possible positive and negative deviations from 4 as 4^{+1}_{-1}.

For more complicated intervals, a midpoint may not be the best choice. For example, for an interval $[1.95, 2.1]$, the natural representative is 2, so the natural representation of this interval is $2^{+0.1}_{-0.05}$. By a "natural representation" we mean that if, say, a physicist tries to measure an unknown quantity y, and as a result of the measurement, he gets the interval $[1.95, 2.1]$ of possible values, then, most probably, he will publish this result as $y \approx 2$. The reason for choosing 2 is that the hypothesis $y = 2$ seems to be the *simplest possible hypothesis*, i.e., 2 *seems to be the simplest possible number from this interval.*

This "simplest" number is not always an integer or a rational number: e.g., for an interval $[3.141592, 3.141593]$, the natural representative is, most likely, π.

The natural question is: is it possible to design an algorithm that would, given a set of real numbers, choose its simplest element?

2 How to Formalize "The Simplest"?

To answer this question, we must first define what "simple" means. Intuitively, a number is simple if it is easy to describe; in other words, a number is simple if the length of its description is small. It is natural to use this intuitive idea to give a precise definition.

Traditionally, foundations of mathematics are based on set theory. So, in principle, we can fix some standard version of set theory (e.g., Zermelo-Fraenkel theory ZF), and consider definitions within this theory. Such a formalization

would have made the definitions (and maybe proofs) slightly shorter. However, these shorter-to-prove results will only apply to ZF. What is we use another version of set theory? What if we use alternative foundations of mathematics (e.g., based on categories instead of sets)? We will see that our results do not depend on which version of set theory we choose, and do not even depend on whether we use set theory or any other formalization of mathematics. To convey this generality, we will formulate our results in the most general form.

We want to be able to have variables that run over real numbers (i.e., whose possible values are real numbers), variables that run over integers, and maybe some other types of variables (e.g., variables that run over intervals, and/or variables that run over arbitrary sets). So, the natural language to use is *multi-sorted first order logic*. For the convenience of the readers who may not be well familiar with this notion, let us give sketchy definitions here; readers who are interested in technical details can look, e.g., in [1, 9, 19].

Definition 1.

- *Let a finite set A be fixed. This set will be called an alphabet, and elements of this set will be called symbols. We assume that this set does not contain symbols $(,), \&, \vee, \neg, \rightarrow, \forall, \exists$, and symbols with subscripts.*
- *By a multi-sorted first order language, we mean the tuple $L = (\mathcal{S}, \mathcal{P}, \mathcal{F}, ar)$, where*
  - *$\mathcal{S}$, $\mathcal{P}$, and $\mathcal{F}$ are subsets of the set A that have no common elements (i.e., $\mathcal{S} \cap \mathcal{P} = \mathcal{S} \cap \mathcal{F} = \mathcal{P} \cap \mathcal{F} = \emptyset$);*
    - *elements of the set $\mathcal{S}$ will be called sorts;*
    - *elements of the set $\mathcal{P}$ will be called predicate symbols;*
    - *elements of the set $\mathcal{F}$ will be called function symbols.*
  - *ar is a function that transforms every elements from the set $\mathcal{P} \cup \mathcal{F}$ into a non-empty finite sequence of sorts (i.e., of elements of $\mathcal{S}$).*
  - *for every predicate symbol $P \in \mathcal{P}$, the number of elements in a sequence $ar(P)$ is called the arity of this predicate; if this number is 1, the predicate is called unary; if it is 2, the predicate is called binary, etc.;*
  - *for every function symbol $f \in \mathcal{F}$, its arity is defined as the number of elements in $ar(f)$ minus 1; the last symbol is the sequence $ar(f)$ is called its output type; 0-ary functions are called constants of this output type.*
- *Let $s \in \mathcal{S}$ be a sort. By a variable of sort s, we mean an expression of the type x_n^s, where n is a natural number.*
- *The notion of the term and its type is defined as follows:*

- *every variable x_n^s is a term of type s;*
- *if $t_1, \ldots, t_m$ are terms of types $s_1, \ldots, s_m$, and if $f \in \mathcal{F}$ is a function symbol for whom $ar(f) = s_1 \ldots s_m s$, then the expression $f(t_1, \ldots, t_m)$ is a term of type s.*

- *The notion of an elementary formula is defined as follows: If $t_1, \ldots, t_m$ are terms of types $s_1, \ldots, s_m$, and $P \in \mathcal{P}$ is a predicate for which $ar(P) = s_1 \ldots s_m$, then $P(t_1, \ldots, t_m)$ is an elementary formula.*

- *The notion of a formula is defined as follows:*
  - *Every elementary formula is a formula.*
  - *If F and G are formulas, then the expressions (F), $F\&G$, $F \vee G$, $\neg F$, and $F \to G$ are formulas.*
  - *If F is a formula and v is a variable, then expressions $\forall v\, F$ and $\exists v\, F$ are formulas.*

In a standard manner, we can now define *closed* formulas, formulas with one free variable, etc.

Comment. In the following text, we will consider languages in which the list of sorts $\mathcal{S}$ contains two symbols: "integer" and "real", and which contain standard arithmetic predicates and function symbols such as 0, 1, $+$, $-$, $\cdot$, $/$, $=$, $<$, $\leq$, both for integers and for reals.

For reader's convenience:

- we will denote variables that run over real numbers by $x, y, \ldots$ (instead of $x_1^{\text{"real"}}, x_2^{\text{"real"}}, \ldots$), and, correspondingly, variables that run over natural numbers by $m, n, \ldots$; and
- for terms containing standard functions like $+$, we will use traditional notations $x + y$ (with a function symbol inside the expression) instead of a more precise expression $+(x, y)$ following from Definition 1.

Definition 2. *Let a (multi-sorted first order) language L be fixed. By a theory T, we mean a finite set of closed formulas.*

Comment. In the following text, we will always assume that a theory T is *consistent.*

In a standard (and natural) manner, we can now define the notion of an *interpretation* of a language L, in which, crudely speaking:

- to every sort $s \in \mathcal{S}$, we assign a set (whose elements will be called objects of this sort);
- to every predicate symbol, we assign a predicate;
- to every function symbol, a corresponding function.

For each interpretation, we can then interpret terms and formulas, and we say that an interpretation is a *model* of the theory T if all formulas from T are true in this interpretation.

We say that a formula F is *deducible* from the theory T (and denote it by $T \vdash F$) if this formula F is true in every model of the theory T.

Comment. In the following text, we will assume that a theory T is fixed. We will assume that this theory contains both the standard first order theory of integers (Peano arithmetic [1, 9, 19]) and a standard first order theory of real numbers [21, 20, 3, 8].

As we have already mentioned, one of the possibilities is to consider, as the theory T, axiomatic set theory (e.g., ZF), together with explicit definitions of integers, real numbers, and standard operations and predicates in terms of set theory. In this case, the set of sorts consists of three elements: "integer", "real", and "set". However, as we have also mentioned, our definitions and results apply to other theories as well.

Now, we are ready to define "definability".

Definition 3. *Let a language L (whose set of sorts includes the sort of real numbers) and a theory T be fixed. By a definable real number, we mean a formula $F(y)$ with one free variable for real numbers for which*

$$T \vdash \exists y\, F(y) \,\&\, \forall x\, \forall y\, (F(x) \,\&\, F(y) \to x = y).$$

We will also say that a formula $F(y)$ defines a real number.

Comment. Informally, a real number x_0 is definable if there exists a formula $F(x)$ that is true for this real number x_0 that is not true for any other real number $x \neq x_0$.

Examples.

- A formula $y \cdot y = 1 + 1 \,\&\, y \geq 0$ satisfies the Definition 3; thus, it defines a unique real number $(\sqrt{2})$.
- A formula $\forall x\, (x \cdot y = x + x + x)$ defines a real number 3.
- If the language L contains all symbols for standard mathematical functions, including the sine function sin, and if the theory T contains ZF + definitions of these mathematical functions in terms of set theory, then the formula $\sin(y) = 0 \;\&\; 3 \leq y \leq 4$ defines a real number: actually, this number is π.

Definition 4. *Let $F(y)$ and $F'(y)$ be definable real numbers. We say that they define the same real number if*

$$T \vdash \forall x\, \forall y\, (F(x)\,\&\,F'(y) \to x = y).$$

We say that $F(y)$ and $F'(y)$ define different numbers if

$$T \vdash \forall x\, \forall y\, (F(x)\,\&\,F'(y) \to x \neq y).$$

Definition 5.

- *By a length of a formula F, we mean its total length that is counted as follows:*
  - *every symbol from the alphabet A, every parenthesis $(,)$, and every logical symbol $(\&, \vee, \neg, \to, \forall, \exists)$ is counted as one symbol;*
  - *every variable x_n^s is counted as 2 symbols + as many symbols as there are bits in the binary representation of the integer n.*
- *Let $F(y)$ be a definable real number. By it complexity $D(F)$, we mean the length of the shortest formula that defines the same real number.*

Comments.

- This definition is similar to the so-called *Kolmogorov complexity* $C(x)$ (invented independently by Chaitin, Kolmogorov, and Solomonoff), which is defined as the smallest length of the program that *computes* x (for a current survey on Kolmogorov complexity, see, e.g., [15]). In our case, however, we do not care that much about how to compute: computing 3.141592 may be easier than computing π; we are more interested in how easy it is to *describe* x. Due to this difference, we had to modify Kolmogorov's definition.
- The above-defined complexity of a number depends on the theory. Correspondingly, the choice of the simplest number from an interval may also depend on the theory. For example, if this interval is $[0.0625, 0.15625]$, then we can have at least two different situations:
  - On one hand, if we are preparing a publication, then the simplest element of an interval is, most probably, 0.1, because 0.1 is, probably, the simplest possible decimal number on this interval.
  - On the other hand, if we must choose a number for a further computer processing, it makes more sense to choose a number 1/8 that has the simplest *binary* representation (0.001_2).

  These two different situations correspond to two different theories:

– in the theory that correspond to the first situation, we allow decimal numbers as constants but not binary numbers;

– in the theory that correspond to the second situation, we allow binary numbers as constants but not decimal numbers.

3 First Result: It Is Impossible to Algorithmically Choose the Simplest Element of a Finite Set

Comment. When we say that a real number is given, we mean that we are given a *formula* $F(y)$ that defines this number. So, the question becomes: suppose that we are given several numbers. Can we choose the one with the the smallest complexity? We will prove that the answer is negative even for the simplest sets that consist of two real numbers.

Definition 6. *By the problem of choosing the simplest representative from a finite set, we mean the following problem:*

GIVEN:

- *an integer n, and*
- *a set of n definable real numbers, i.e., n formulas $F_1(y), \ldots, F_n(y)$ that define real numbers.*

FIND:

the value i for which the corresponding real number is the simplest, i.e., for which $D(F_i) = \min(D(F_1), \ldots, D(F_n))$.

PROPOSITION 1. *Even for $n = 2$, no algorithm is possible that, given a finite set with n elements, chooses the simplest representative from this set.*

4 Second Result: It Is Impossible to Algorithmically Choose the Simplest Element of an Interval

Definition 7a. *By a definable interval, we mean a pair of formulas $\underline{F}(y)$ and $\overline{F}(y)$ that define real numbers and for which*

$$T \vdash \forall x\, \forall y (\underline{F}(x)\, \&\, \overline{F}(y) \to x \le y).$$

Definition 7b. *We say that a definable real number $F(y)$ belongs to the definable interval $[\underline{F}(y), \overline{F}(y)]$ if*

$$T \vdash \forall x \forall y \forall z (\underline{F}(x) \,\&\, F(z) \,\&\, \overline{F}(y) \to (x \le z \,\&\, z \le y).$$

Definition 8. *By the problem of choosing the simplest representative from an interval, we mean the following problem:*

GIVEN:

a *definable interval* $[\underline{F}(y), \overline{F}(y)]$.

FIND:

- *the formula $F(y)$ that defines the simplest definable real number from the interval $[\underline{F}(y), \overline{F}(y)]$; and,*
- *in case one of the endpoints $\underline{F}(y), \overline{F}(y)$ is the simplest definable number on this interval, the value $-$ or $+$ indicating, correspondingly, whether the lower endpoint $\underline{F}(y)$ or the upper endpoint $\overline{F}(y)$ is the simplest.*

PROPOSITION 2. *No algorithm is possible that, given a definable interval, would return the simplest representative from this interval.*

Comment. The same impossibility result holds if we fix one of the endpoints. To be more precise, this result holds for *almost all* possible choices of the endpoint ("almost all" in some natural sense).

PROPOSITION 3.

- *If $F(y)$ is the simplest possible definable real number, then:*

  There exists an algorithm that, given any definable real number $F'(y)$ that defines a different number, chooses the simplest representative from the corresponding interval $[F(y), F'(y)]$ or $[F'(y), F(x)]$.

- *If $F(y)$ is not the simplest possible real number, then:*

  No algorithm is possible that, given any definable interval with $F(y)$ as one of the endpoints, would choose the simplest representative from this interval.

Definition 9. *We say that a property $\tilde{P}(x)$ holds for almost all definable real numbers if there exists finitely many definable real numbers $F_1(y), \ldots, F_n(y)$ such that: if the definable number $F(y)$ is different from each of them, then the property $\tilde{P}(x)$ holds for the number that is defined by the formula $F(y)$.*

Comment. Informally, we can say that a property holds for almost all definable real numbers if it holds for all definable real numbers, except, maybe, finitely many of them.

COROLLARY. *For almost all definable real numbers $F(y)$, the following property holds:*

- *No algorithm is possible that, given a definable interval with $F(y)$ as one of its endpoints, would choose the simplest representative from this interval.*

Comment. Similar results are true if we restrict ourselves to intervals in which the given number $F(y)$ is the lower endpoint (or, correspondingly, the upper endpoint).

PROPOSITION 4.

- *For any definable real number $F(y)$, the following two properties are equivalent to each other:*
  - *The number defined by the formula $F(y)$ is the simplest of all definable real numbers that are $\geq$ that this number.*
  - *There exists an algorithm that, given any definable interval with $F(y)$ as its lower endpoint, chooses the simplest representative from this interval.*
- *For any definable real number $F(y)$, the following two properties are equivalent to each other:*
  - *The number defined by the formula $F(y)$ is the simplest of all definable real numbers that are $\leq$ that this number.*
  - *There exists an algorithm that, given any definable interval with $F(y)$ as its upper endpoint, chooses the simplest representative from this interval.*

5 Similar Results Hold for Computable Real Numbers

For the cases when the intervals (from which we are choosing the simplest numbers) come from computations, it is reasonable not to consider arbitrary *definable* real numbers, but to restrict ourselves to *computable* real numbers, i.e., real numbers that can be computed with an arbitrary accuracy (see, e.g., [4, 7, 2, 5]):

Definition 10. *A real number x is called constructive if there exists an algorithm (program) that transforms an arbitrary integer k into a rational number x_k that is 2^{-k}−close to x. It is said that this algorithm computes the real number x.*

Comment. Every constructive real number is uniquely determined by the corresponding algorithm and is, therefore, definable.

Comment. When we say that a constructive real number is given, we mean that we are given an algorithm that computes this real number.

Definition 11. *By the problem of choosing the simplest representative from a constructive interval, we mean the following problem:*

GIVEN:

a constructive interval, i.e., algorithms $\underline{U}$ and $\overline{U}$ that compute real numbers $\underline{x} < \overline{x}$.

FIND:

the simplest (in the sense of $D(F) \to \min$) constructive real number from the interval $[\underline{x}, \overline{x}]$.

PROPOSITION 5. *No algorithm is possible that, given a constructive interval, returns the simplest representative from this interval.*

PROPOSITION 6.

- *If x is the simplest possible constructive real number, then:*

  There exists an algorithm that, given any other constructive real number $y \neq x$, chooses the simplest representative from the corresponding interval $[x, y]$ or $[y, x]$.

- *If x is not the simplest possible constructive real number, then:*

  No algorithm is possible that, given any other constructive real number $y \neq x$, would choose the simplest representative from the corresponding interval $[x, y]$ or $[y, x]$.

PROPOSITION 7.

- *For any constructive real number x, the following two properties are equivalent to each other:*
  - *The number x is the simplest of all constructive real numbers $\geq x$.*
  - *There exists an algorithm that, given any constructive real number $y > x$, chooses the simplest constructive real number from the interval $[x, y]$.*
- *For any definable real number x, the following two properties are equivalent to each other:*
  - *The number x is the simplest of all constructive real numbers $\leq x$.*
  - *There exists an algorithm that, given any constructive real number $y < x$, chooses the simplest constructive real number from the interval $[y, x]$.*

6 Proofs

General comment. The results of this paper are mainly based on results from mathematical logic.

6.1 Proof of Proposition 1

We will prove our result by reduction to a contradiction. Let us assume that there exists an algorithm U that for every two defining properties F_1 and F_2 tells whether the first or the second one defines the simplest number.

Since we have assumed, in effect, that the theory T contains formal (= first order) arithmetic, we can use the famous Gödel's theorem and conclude that this theory is *undecidable*, i.e., that there exists no algorithm that, given a formula F from this languages, would tell whether this formula is deducible from T or not (see, e.g., [1, 9, 19]).

Moreover, no algorithm is possible, that is applicable to an arbitrary arithmetic formula F and that would return "yes" if F is deducible from T and "no" if the negation $\neg F$ of the formula F is deducible from T (see, e.g., [19], Chapter 6, Ex. 13(c)); see also [18], Sections 7.7–7.9). We will show that our hypothetic algorithm U leads exactly to such an impossible algorithm.

Indeed, let us take an arbitrary definable number and denote it by F^-.

Since the language L contains the formal arithmetic, all integers are defined in this language. Therefore, there are infinitely many definable numbers. Since for every length l, there are only finitely many formulas of this length, these formulas can only define finitely many different numbers. Thus, for every length l, there exists a definable number that cannot be defined by any formula of length

$\leq l$, and for which, therefore, the complexity is $> l$. In particular, there exists a definable number whose complexity is greater than $l = D(F^-)$. Let us pick one such number and denote it by $\tilde{F}(y)$. Similarly, there exists a definable number whose complexity is $> D(\tilde{F})$. Let us pick one such number and denote it by F^+.

So, we have three definable numbers $F^-(y)$, $\tilde{F}(y)$, and $F^+(y)$, for which $D(F^-) < D(\tilde{F}) < D(F^+)$.

Let us now consider the following formula:

$$(F \to F^-(y))\&(\neg F \to F^+(y)).$$

We will denote this formula by $F'(y)$. Let us first prove that this formula indeed defines a real number:

- if F is deducible from T, then this formula $F'(y)$ clearly defines a real number (namely, the same real number as $F^-(y)$);
- similarly, if $\neg F$ is deducible from T, then this formula $F'(y)$ also defines a real number (namely, the same real number as $F^-(y)$);
- since in classical logic, we have $T \vdash F \vee \neg F$, we can thus conclude that this formula *always* defines a real number.

In particular, if either F or $\neg F$ is deducible from the theory T, then:

- If F is deducible from T, then this new formula is equivalent to $F^-(y)$ and therefore, it defines the same number as $F^-(y)$.
- If F is not deducible from T, then this formula is equivalent to $F^+(y)$ and thus, it defines the same number as $F^+(y)$.

Let us now apply our hypothetic algorithm U to the formulas $F'(y)$ and $\tilde{F}(y)$:

- If F is deducible from T, then U will select the number defined by $F'(y)$, because this number ($F^-(y)$) is simpler than the number defined by $\tilde{F}(y)$.
- If $\neg F$ is deducible from T, then U will select the number defined by the formula $\tilde{F}(y)$, because in this case, this number is simpler than the number ($F^+(y)$) defined by the formula $F'(y)$.

Thus, simply by looking at the output of the algorithm U, we get an algorithm that returns "yes" if F is deducible from T and "no" if its negation $\neg F$ is deducible from T. We already know that such an algorithm is impossible.

This contradiction shows that our initial assumption — that the problem of choosing the representative from a finite set is algorithmically solvable — is false. Hence, this problem is not algorithmically solvable. Proposition 1 is proven.

Comments.

- This result is similar to the known result that Kolmogorov complexity is not decidable [15]. In effect, our result sounds slightly stronger because we have proven that not only computing the actual values of complexity is impossible, but even deciding which of the values has larger complexity is also impossible.
- Simplicity seems to be a natural criterion for choosing a representative, but we can also look for other ways in which a number can be representative. For example, we may want a number that is the most "typical" of the elements of the given set; this approach is outlined, for different notions of "typicality", in [11, 12, 13, 10, 6].

6.2 Proof of Propositions 2–4

Let us first prove Proposition 3. Then, we will show that the Corollary (and hence, Proposition 2) is also true.

6.2.1 Case of the Simplest Possible Definable Real Number $F(y)$

Let $F(y)$ be the simplest possible definable real number. This means that its complexity $D(F)$ is the smallest possible complexity that a real number can have. In other words, the number $F(y)$ is defined by a formula of length $l_{\min}$ that is the shortest possible formula defining a real number. For such $F(y)$, it is easy to describe the desired algorithm: from every interval $[F(y), F'(y)]$ or $[F'(y), F(y)]$ that has $F(y)$ as one of its endpoints, we can return this very definable number $F(y)$ as the desired simplest representative.

6.2.2 Case of a Definable Real Number $F(y)$ That Is Not the Simplest Possible

Let now $F(y)$ be *not* the simplest possible real number. For such $F(y)$, we will prove the impossibility of an algorithm by reduction to a contradiction. Let us assume that there exists an algorithm U that, given any other definable real number $F'(y)$:

- chooses the simplest representative s from the corresponding interval $[F(y), F'(y)]$ or $[F'(y), F(y)]$; and
- if this simplest representative coincides with one of the endpoints, returns $-$ or $+$ depending on whether s is the left or the right endpoint.

The fact that $F(y)$ is not the simplest possible number means that there exist other definable real numbers whose complexity is smaller than $D(F)$, i.e., that are defined by formulas shorter than $D(F)$. We have already shown in the proof

of Proposition 1 that for every length l, there exist finitely many definable real numbers of complexity l. Thus, there exist finitely many definable real numbers that are simpler than $F(y)$. From these numbers, let us pick the formula $G(y)$ for which the number defined by it is the closest to $F(y)$ (if there are two such numbers, let us pick the one that is greater than the number defined by $F(y)$).

Without loss of generality, we can assume that the number x_F defined by the formula $F(y)$ is smaller that the number x_G defined by the formula $G(y)$ (the case $x_G < x_F$ can be considered similarly). Now, let $f(n)$ be any algorithmic function from natural numbers to natural numbers. It is known that every algorithmic sequence is definable in Peano arithmetic, and therefore, since out theory T includes Peano arithmetic, $f(n)$ is definable in T as well.

For every such function, we can define a new definable number z_f as follows:

- If $\forall n(f(n) = 0)$, then $z_f = x_G$.
- If $\exists n(f(n) \neq 0)$, then $z_f = x_G - 2^{-n_{\min}} \cdot (x_G - x_F)$, where $n_{\min}$ is the smallest natural number n for which $f(n) \neq 0$.

(We have used words to define z_f, but this definition can be easily reformulated in terms of formulas, so, the number z_f is indeed definable.)

For each function f, it is easy to see which element from the interval $[x_F, z_f]$ is the simplest:

- If $\exists n(f(n) \neq 0)$, then $x_F < z_f < x_G$. Since we have chosen x_G as the closest of all definable real numbers that are simpler than x_F, and since all the elements of the semi-open interval $(x_F, z_f]$ are closer to x_F than x_G, we can conclude that none of the real numbers from the interval $(x_F, z_f]$ is simpler than x_F. Thus, x_F is the simplest of all real numbers from the interval $[x_F, z_f]$.
- If $\forall n(f(n) = 0)$, then $z_f = x_G$. Since we have chosen x_G as the closest of all definable real numbers that are simpler than x_F, and since all the elements of the open interval (x_F, x_G) are closer to x_F than x_G, we can conclude that none of the real numbers from the open interval (x_F, x_G) is simpler than x_F. Thus, x_G is the simplest of all real numbers from the interval $[x_F, x_G] = [x_F, z_f]$.

In both cases, the simplest element coincides with one of the endpoints, so, the algorithm U will return either $-$ or $+$:

- If $\exists n(f(n) \neq 0)$, then the lower endpoint (x_F) is the simplest, and hence, the algorithm U will return $-$.
- If $\forall n(f(n) = 0)$, then the upper endpoint (z_f) is the simplest, and hence, the algorithm U will return $+$.

Thus, by checking whether the sign returned by the algorithm U is $-$ or $+$, we will be able to check, for a given computable function f, whether $\forall n(f(n) = 0)$ is true or not.

However, it is known (see, e.g., [14, 16, 17]) that there exists *no* algorithm for deciding whether a program (to be more precise, a program that always finishes its computations) always returns 0. In other words, there exists no algorithm, that, given an algorithmic (everywhere defined) function $f(n)$ from natural numbers to natural numbers would check whether $\forall n(f(n) = 0)$. This contradiction shows that our initial assumption — that the problem of choosing the representative from an interval is algorithmically solvable — is false. Hence, this problem is not algorithmically solvable. Proposition 3 is proven.

6.2.3 Proof of the Corollary

Let us now prove the Corollary (and thus, Proposition 2). In the proof of Proposition 1, we have already shown that for every length l, there exist finitely many definable real numbers of complexity l. In particular, this means that there exist finitely many definable real numbers of the smallest possible complexity $l_{\min}$. Thus, every property (including the property $*$) that holds for all definable real numbers, except for the simplest ones, is thus true for almost all definable real numbers. Corollary is proven. Q.E.D.

6.2.4 Proofs of Proposition 4

Proposition 4 can be proven similarly to the proof of Proposition 3.

6.3 Proof of Propositions 5–7

If x is the simplest possible constructive real number, then we can always return x.

If x is not the simplest possible constructive real number, then we can use the same construction as in the proof of Proposition 3. To complete the proof, we must now prove only the following two additional statements:

- First, we need to prove that z_f is a constructive real number (and that, given a program f, we can construct a program (algorithm) for computing z_f).
- Second, in our definition, we no longer require the algorithm to return $-$ or $+$. Therefore, to complete the proof, we must show that if an algorithm returns a constructive real number s that is equal to one of the endpoints (i.e., to x or to z_f), then we can algorithmically check whether this constructive real number coincides with the left or with the right endpoint.

Both statements are (relatively) easy to prove:

- To compute z_f with an accuracy $(z-x)\cdot 2^{-k}$, it is sufficient to compute first k values of f, and take:
  - If $\forall n_{\le k}(f(n)=0)$, then $a_k = z$.
  - If $\exists n_{n\le k}(f(n)\neq 0)$, then $a_k = z - 2^{-n_{\min}}\cdot(z-x)$, where $n_{\min}$ is the smallest natural number $n \le k$ for which $f(n)\neq 0$.

  Then, as one can easily see, $|a_k - z_f| \le 2^{-k}\cdot|z_f - x| \le 2^{-k}\cdot|z-x|$. From these values a_k, we can easily compute the desired rational approximations z_{fk} to z_f.

- If an algorithm returns a constructive real number s that coincides with one of the constructive endpoints of the interval $[x, z_f]$, then, by computing x, z_f, and s with sufficient accuracy (namely, with accuracy $\varepsilon < (z_f - x)/4$), and comparing the corresponding rational numbers, we will be able to check whether $s = x$ or $s = z_f$. Indeed, in this case, from $|s_k - s| \le \varepsilon$, and $|z_{fk} - z_f| \le \varepsilon$, we can conclude that

$$|z_{fk} - s_k| \ge |z_f - s| - |s_k - s| - |z_{fk} - z_f| >$$

$$|z_f - s| - 2\cdot(1/4)\cdot|z_f - s| > (1/2)\cdot|z_f - x|.$$

  Hence:
  - If $s = x$, then, similarly, $|s_k - z_{fk}| > (1/2)\cdot|z_f - x|$. On the other hand, in this case, $|s_k - x_k| \le |s_k - s| + |x_k - x| \le 2\varepsilon < (1/2)\cdot(z_f - x)$. Therefore, in this case, $|s_k - x_k| < |s_k - z_{fk}|$.
  - Similarly, if $s = z_f$, then $|s_k - x_k| > |s_k - z_{fk}|$.

  Thus, comparing two rational numbers $|s_k - x_k|$ and $|s_k - z_{fk}|$, we can tell with which of the endpoints s coincides.

Q.E.D.

Acknowledgments. This work was partially supported by NASA Grants No. NAG 9-757, NCC5-209, and NCC 2-1232, by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-00-1-0365, by NSF grants CDA-9522207 and 9710940 Mexico/Conacyt, and by Grant No. W-00016 from the U.S.-Czech Science and Technology Joint Fund.

This work was partially done when V.K. was an invited professor at Laforia, University of Paris VI, and partially done during V.K.'s visit to Wuppertal. The authors are thankful to Bernadette Bouchon-Meunier and to the anonymous referees for valuable comments and help.

References

[1] J. Barwise (ed.). *Handbook of Mathematical Logic.* North-Holland, Amsterdam, 1977.

[2] M. J. Beeson, *Foundations of constructive mathematics*, Springer-Verlag, N.Y., 1985.

[3] N. Ben-Or, D. Kozen, and J. Reif, "The complexity of elementary algebra and geometry", *Journal of Computer and System Sciences*, 1986, Vol. 32, pp. 251–264.

[4] E. Bishop, *Foundations of Constructive Analysis*, McGraw-Hill, 1967.

[5] E. Bishop, D. S. Bridges, *Constructive Analysis*, Springer, N.Y., 1985.

[6] B. Bouchon-Meunier, M. Rifqi, and S. Bothorel, "Towards general measures of comparison of objects", *Fuzzy Sets and Systems*, 1996 (to appear).

[7] D. S. Bridges, *Constructive Functional Analysis*, Pitman, London, 1979.

[8] J. Canny, "Improved algorithms for sign determination and existential quantifier elimination", *The Computer Journal*, 1993, Vol. 36, No. 5, pp. 409–418.

[9] H. B. Enderton. *A mathematical introduction to logic.* Academic Press, N.Y., 1972.

[10] M. Friedman, M. Ming, and A. Kandel, "On the theory of typicality", *International Journal of Uncertainty, Fuzziness, and Knowledge-Based Systems*, 1995, Vol. 3, No. 2, pp. 127–142.

[11] G. Heindl and E. Reinhart, "Adjustment by the principle of minimal maximum error", In: *Beiträge aus der Bundesrepublik Deutschland zur Vorlage bei der XVI Generalversammlung der Internationalen Union für Geodäsie und Geophysik, Grenoble 1975*, Veröffentlichungen der Deutschen Geodätischen Kommission, Reihe B, München, 1975, Vol. 213, pp. 33–43.

[12] G. Heindl and E. Reinhart, *Ausgleichung im Sinne minimaler Maximalfehler*, Veröffentlichungen der Deutschen Geodätischen Kommission bei der Bayerischen Akademie der Wissenschaften, Reihe A, München, 1976, Vol. 84.

[13] G. Heindl and E. Reinhart, "Experience with a non-statistical method of directing outliers", In: *Proceedings of the International Symposium on Geodetic Networks and Computations of the International Association of Geodesy, Volume V, Network Analysis Models*, Veröffentlichungen der Deutschen Geodätischen Kommission bei der Bayerischen Akademie der Wissenschaften, Reihe B, München, 1982, Vol. 258/V, pp. 19–28.

[14] L. R. Lewis and C. H. Papadimitriou, *Elements of the theory of computation*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[15] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, N.Y., 1997.

[16] J. C. Martin, *Introduction to languages and the theory of computation*, McGraw-Hill, N.Y., 1991.

[17] C. H. Papadimitriou, *Computational Complexity*, Addison Wesley, San Diego, 1994.

[18] H. Rogers. Jr., *Theory of recursive functions and effective computability*, McGraw-Hill, N.Y., 1967.

[19] J. R. Schoenfield. *Mathematical logic*. Addison-Wesley, 1967.

[20] A. Seidenberg, "A new decision method for elementary algebra", *Annals of Math.*, 1954, Vol. 60, pp. 365–374.

[21] A. Tarski, *A Decision Method for Elementary Algebra and Geometry*, University of California Press, Berkeley, 1948.