Notre Dame Journal of Formal Logic
Volume XX, Number 1, January 1979
NDJFAM

A SIMPLE GENERALIZATION OF TURING COMPUTABILITY

WILLIAM J. THOMAS

1 *Introduction* In the present paper,* the notion of a Turing machine is generalized as follows: the notion of an *ideal computation* is defined to be a finite sequence of expressions of some arbitrary language. An *ideal machine* is then defined to be a set of *ideal computations* such that functionality from first to last elements of the *ideal computations* in the *ideal machine* obtains. Then, using the devices of Gödel numbering, a *length* function (applied to *ideal computations*) and *input* and *output* functions (which, when applied to *ideal computations*, select their first and last elements, respectively), a generalized analogue of the Kleene T-predicate is defined. A class of *ideal machines* is then said to be an **R**-class just in case it is Gödel numerable and contains an *ideal machine* which, in a sense made precise, *decides* the generalized T-predicate analogue.

The following results will be obtained: (1) The classes of Turing machine computations and Post normal system proofs are **R**-classes. (2) There exist **R**-classes which properly include the class of Turing machines. (3) For any **M**, where **M** is an **R**-class, there exist functions which are not computable by **M**. This theorem generalizes the Turing unsolvability of the "halting problem", and its proof makes use of the familiar diagonal procedure. Defining **M**-decidability for **R**-classes **M** in the natural way, we have that: (4) In general, the class of **M**-decidable sets is not closed under Boolean operations on pairs of sets. (5) There exist trivial **R**-classes, indeed **R**-classes which contain only one *ideal machine*. (6) In general, **R**-classes are not invariant with respect to Gödel numbering. (7) Mostowski's generalization of recursive function theory, $\mathfrak{R}$-definability (developed in [1]), is properly included in the notion of **R**-ness.

---

*Presented to the 1973-74 Annual Meeting of the Association for Symbolic Logic, Atlanta, Georgia, December 27-28, 1973. This paper is based on material included in Chapter V of my Ph.D. Dissertation *Church's Thesis and Philosophy*. Special thanks are due to my advisors, Professor Raymond J. Nelson and Professor Howard Stein.

Received January 3, 1974

Finally, the philosophical significance of these results will be discussed. It will be observed that, for certain purposes, a hypothesis (*viz.* that any formal reconstruction of the notion of effective computation can be cast as an **R**-class) weaker than the conjecture that all computable functions are recursive suffices. It will be argued that this weaker hypothesis strains our intuitions less than does Church's Thesis. An optimistic conjecture about the pedagogical usefulness of the simple generalization will be offered.

2. *Mathematical development and results* It should be understood at the beginning that what we shall be formalizing here are some necessary conditions for an adequate theory of effectiveness. Our generalization is demonstrably not sufficient for such a theory.

Definition 1 By an *ideal computation* (**i.c.**) we understand a sequence $(i_k, e_1, \ldots, e_n, o_j)$, where the e_i are understood to be expressions of some language $\mathcal{L}$, i_k and o_j are elements of $\mathcal{L}' \subseteq \mathcal{L}$, such that there exists a function f: $\mathcal{L}' \overset{1-1}{\underset{\text{onto}}{\rightarrow}} N$, (where N is the set of natural numbers). We shall, for convenience, take the elements of $\mathcal{L}'$ to be numerical representations of natural numbers, where $f(e_i) = n$ *iff* e_i represents n.

A "language" is understood here in the familiar way as some chosen subset of the set of all finite sequences on some chosen alphabet (set of symbols). In what follows, what is important about the sequence $(e_1, \ldots, e_n)$ is just its ordinal length. Thus the particular language chosen is of no consequence for our concerns here. The sequence $(e_1, \ldots, e_n)$ is intended to generalize the evolution of a formal computation, or, when the theory we are developing is applied to formal proofs, rather than computations, $(e_1, \ldots, e_n)$ generalizes the notion of a proof. It will of course be noticed that there is no requirement that the sequence $(e_1, \ldots, e_n)$ be effective, whereas an actual formal computation will be an effective sequence, in that the transition function from the e_i's up to k, to e_k will, in the case of an actual computation, be effective. All that is asked of the reader's intuitions at this juncture is an acceptance of the claim that all computations are **i.c.**'s.

Definition 2 An *ideal machine* (**i.m.**) is a set **M** of **i.c.**'s such that:

(a) If $(i_k, e_g, \ldots, e_n, o_j) \in \mathbf{M}$ and $(i_k, e_h, \ldots, e_i, o_1) \in \mathbf{M}$ then $o_j = o_1$. (This requirement might be termed the "monogenicity requirement".)
(b) If $(i_k, e_m, \ldots, e_n, o_j) \in \mathbf{M}$ and $(i_k, e_i, \ldots, e_h, o_j) \in \mathbf{M}$ then $(e_m, \ldots, e_n) = (e_i, \ldots, e_h)$. (This requirement may be termed the "determinacy of operation" requirement.)
(c) The e_i of the **i.c.**'s belonging to **M** all belong to some finite set of expressions.

The "determinacy of operation" requirement (b) may be dropped and the rest of the theory modified slightly, giving a nice generalization of non-deterministic machines. To do so complicates matters somewhat, and is not germane to our present project, and so the development will be postponed.

Definition 3 The input function, i, is defined: $\mathrm{i}(i_k, e_1, \ldots, e_n, o_j) = i_k$, and the output function, o is: $\mathrm{o}(i_k, e_1, \ldots, e_n, o_j) = o_j$.

Definition 4 The length function l, applied to $(i_k, e_m, \ldots, e_n, o_j)$ is just the length of the sequence $(e_m, \ldots, e_n)$.

Definition 5 Let $\mathrm{g}'(x, y, z)$ be some standard Gödel-numbering of ordered triples of natural numbers.

Definition 6 A class **M** of **i.m.**'s is an **R**-class *iff*:

(a) There exists a map g: $\mathbf{M} \overset{1\text{-}1}{\rightarrow} N$, (so that each **i.m.** in **M** has a numerical name, called that machine's *index*. We will denote the **i.m.** having index i by $\mathbf{M}_i$.)

and,

(b) (For convenience, let us write '$T_{\mathbf{M}}(x, y, z)$' to indicate that $(Eu)[u \in \mathbf{M} \,\&\, \mathrm{g}(u) = x \,\&\, (Ew)(w \in u \,\&\, \mathrm{i}(w) = y \,\&\, \mathrm{l}(w) = z)]$. $T_{\mathbf{M}}$ is an analogue, generalized to **M**, of the Kleene T-predicate.)

$$(Eu)(x)(y)(z)\{u \in \mathbf{M} \,\&\, [T_{\mathbf{M}}(x, y, z) \rightarrow (Es)(s \in u \,\&\, \mathrm{i}(s) < \mathrm{g}'(x, y, z) \,\&\, \mathrm{o}(s) = 1)] \,\&\, [T_{\mathbf{M}}(x, y, z) \rightarrow (Es)(s \in u \,\&\, \mathrm{i}(s) = \mathrm{g}'(x, y, z) \,\&\, \mathrm{o}(s) = 0)]\}$$

This condition amounts to a provision that **R**-classes must be Gödel numerable, and that they must contain a machine which is a generalized analogue of the Turing machine which decides the Kleene T-predicate.

Theorem 1 *The classes of Turing machine computations and Post normal systems proofs are* **R**-*classes.*

Proof: Let **M** be the class of Turing machine computations (or Post normal system proofs). **M** is Gödel numerable, by assigning the Gödel number (under some standard system of Gödel numbering) of each Turing machine (formulated in one of the standard ways, say as a set of quadruples) to its associated set of computations. Thus condition (a) of Definition 6 is satisfied. It is a basic theorem of recursive function theory that $T_{\mathbf{M}}$ is recursively decidable, and so condition (b) is satisfied.

Theorem 1 suggests a construal of Turing machines and other known formalized conceptions of algorithms as sets of computations, or algorithmic procedures. So construed, Turing machines, and other equivalent concepts, constitute **R**-classes. In the results that follow, we shall speak of Turing machines as sets of **i.c.**'s.

Theorem 2 *There exist* **R**-*classes which properly include the class of Turing machines.*

Proof: Let **M** be the union of the set of Turing machines (construed as sets of **i.c.**'s) with the unit set containing $\overline{R}$, a set of triples $\langle x, e, y\rangle$, where e is some arbitrarily chosen constant expression, and where, for some chosen non-recursive total function f, $f(x) = y$. It is obvious that **M** properly

includes the class of Turing machines. It remains only to show that **M** is an **R** class.

The class **M** is certainly a class of **i.m.**'s. Let the index of $\overline{R}$ be 0, and let the index of each Turing machine be just its Gödel number, as in Theorem 1. The result is a map from **M** to N. We now need to establish that condition (b) of Definition 6 obtains of **M**. We shall argue informally that a u of the required kind belongs to the class of Turing machines, and so to the union of that class with $\overline{R}$. Consider an arbitary triple $\langle x, y, z\rangle$ of natural numbers. In the cases where $x \neq 0$, the Turing machine which decides the T-predicate will decide $T_{\mathbf{M}}(x, y, z)$. In case $x = 0$, the required machine must simply decide whether $z = 1$, producing a '1' if it does, and a '0' otherwise. It is clear that there is a Turing machine which decides whether or not $x = 0$, and then, if it does not, mimics the T-predicate deciding machine, and if it does, decides whether $z = 1$ or not.

Definition 7 We say that a function f is **M**-computable just in case:

$$(\mathrm{E}m)(x)\,[m \in \mathbf{M}\ \&\ (x \in \mathrm{dom}(f) \rightarrow (\mathrm{E}y)(y \in m\ \&\ \mathrm{i}(y) = x\ \&\ \mathrm{o}(y) = f(x)))]$$

Definition 7 is a straightforward generalization of the standard notion of Turing computability.

Theorem 3 *For any* **M**, *where* **M** *is an* **R**-*set, there exist functions which are not* **M**-*computable.*

Proof: Let **M** be an **R** set. It is a consequence of Definition 2 that the relation consisting of all and only the inputs with the outputs of the **i.c.**'s in an **i.m.** is a partial function. By Definition 6, the **i.m.**'s in **M** have indices; let us call those **i.m.**'s in **M** M_i. Let us call the function consisting of all and only the inputs with the outputs of the **i.c.**'s in M_i ϕ_i. We shall say that M_i computes ϕ_i. Let:

$$\theta(y) = \phi(y) + 1 \text{ if } (\mathrm{E}x)(T_{\mathbf{M}}(y, y, x)), \text{ and } = 0 \text{ otherwise.}$$

Then θ is not **M**-computable. Suppose, with a view to *reductio*, that θ is **M**-computable. Then, by Definition 7, there exists an $M_i \in \mathbf{M}$ which computes θ. Let us call that M_i M_q. Then:

(i) $(x)(\theta(x) = \phi_q(x))$

But substituting for x in (i), in accord with the definition of θ, we have:

(ii) $\theta(q) = \phi_q(q)$

But, on the hypothesis that M_q *computes* θ, we have:

(iii) $\theta(q) = \phi_q(q) + 1$

The result of conjoining (ii) and (iii) is a contradiction. Therefore, by *reductio ad absurdum*, θ is not **M**-computable. Such are the hazards.

Theorem 3 generalizes the unsolvability of the "halting problem". The diagonal procedures used in the proof of the unsolvability of the

"halting problem" and in the proof of Theorem 3 are essentially the same. Again, defining **M**-decidability and **M**-enumerability in ways analogous with the standard definitions of the corresponding recursive function theoretic notions, our Theorem provides examples of **M**-enumerable but not **M**-decidable sets, where **M** is any **R** class. While we shall want to argue (in section **3**) that the notion of **R**-ness is a necessary condition for a formal reconstruction of effectiveness, it is easily shown that, in general, **R**-sets lack some properties which we would expect the class of algorithms to have. The basic theorem is this:

Theorem 4 *There exist* **R**-*classes* **M** *each that the class of functions* ϕ_i *associated with the* M_i *of* **M** *is not closed under addition of a constant.*

Proof: Consider the **M** constructed in the proof of Theorem 2. It is clear that $f + 2 \neq \phi_i$, for all i, where $M_i \in \mathbf{M}$.

Beginning with this result, it is easily shown that, where **M** is an **R**-class, the class of **M**-decidable sets need not be closed under Boolean operations on pairs of sets. There is too little structure imposed by the weak conditions of Definition 6 to give rise to a really interesting theory. There exist trivial **R**-sets:

Theorem 5 *There exist* **R**-*sets which contain only one* **i.m.**

Proof: Consider the **M** which contains as its sole element the M which computes the function $f(x) = 1$ if $x = 1$, and 0 otherwise. All **i.c.**'s in M are of length 1. Set g such that $g(M) = 1$, g' such that $g' \langle 1, 1, 1 \rangle = 1$, and **M** may be seen to be an **R**-set. **R**-sets disjoint from **M** can be constructed from **M** by substituting a numeral denoting n ($n \neq 1$) for each occurrence of '1' in the description of M.

R-sets are not invariant with respect to Gödel numberings (i.e., the functions g and g'). Most interesting closure properties are lacking, in fact, we have:

Theorem 6 $\mathfrak{R}$-*definability is properly included in the notion of* **R**-*ness.*

Proof: Systems of equations are **i.c.**'s having the properties required by Definition 6. The example used in the proof of Theorem 2 suffices to show that the inclusion is proper.

On the other hand, there are some obvious analogies between the theory of **R**-sets and the theory of recursive functions which might be interesting to study. One might, for example, try to construct an **R** generalization of the Kleene hierarchy, or a theory of degrees of **R**-undecidability.

The theory of **R**-ness, as developed here, depends heavily on standard recursive function theoretic results, in that we have, in our proofs, made heavy use of such results. It would be interesting to explore the logical relationship between **R**-ness and general recursiveness much more deeply than we have done here. Our Theorems 1 and 2 answer only the most obvious of the natural questions about this. For example, it would remove

most of the purely mathematical interest which **R**-ness might have if it should turn out that the **R**-sets are familiar pieces of furniture in the recursive function theorist's universe. Supposing no such result as that to be forthcoming, it would be good to have some more results about which recursive function theoretic entities are **R**-classes. It could be shown, for example, that degrees of recursive unsolvability are **R**-classes, when degrees are construed analogously to the construal of Turing machines in Theorem 1. (Construe a degree as a class of Turing "oracle" machines, and an "oracle" machine as a class of **i.c.**'s.)

3 *Philosophical conclusions* Church's Thesis ('**CT**' hereinafter), being a biconditional, may be thought of as a conjunction of two conditionals. I shall now argue that our generalization of the basic part of elementary recursive function theory provides some of those philosophical consequences of **CT** sometimes called "unfortunate" without burdening our intuitions to the extent that **CT** does. Let us focus attention on the following thesis:

RT: *The set of all effectively computable functions of natural numbers is an* **R**-*class.*

RT is analogous to (and is a generalization of) the "problematic" conjunct of **CT**. By our Theorem 1, if the "problematic" conjunct of **CT** is true, then **RT** is true. Thus, whatever basis one has for believing **CT** is also a basis for accepting **RT**. Further, our Theorem 2 shows that **RT** is, in a certain formal sense, a weaker hypothesis than the "problematic" conjunct of **CT**.

I have argued elsewhere (in [2]) that there are no very good reasons known for believing the "problematic" conjunct of **CT**. I think that there are good reasons for believing **RT**, though not compelling ones. Essentially just two propositions must be accepted: that something like a Gödel numbering will be available for any formal reconstruction of effective computability that we would seriously entertain, and the somewhat more complex assertion that there ought to be an algorithm which decides whether an expression is representative of an algorithm, and if it is, whether the function computed by that algorithm is defined at a certain argument or not, and if it is, whether a given number represents the length of that computation or not.

In recursive function theory, as given in terms of various notions (Turing machines, λ-definability, normal systems, or whatever), algorithms are regarded as sets of expressions which may be thought of as *instructions* for dealing with various inputs. In our generalization, there is nothing that really corresponds to the intuitive idea of an instruction. The notion of **R**-ness is, however, a notion of higher *type* than any specific reconstruction of the notion of effective computability that one might wish to consider. Our generalization does not deal with how one might actually describe an algorithm, or **i.m.** But surely one thing that we require of a formal notion of an algorithm is that we be instructed as to how computations are to be

carried out. That would seem to entail each algorithm *corresponding to* (if not *being composed of*) a set of instructions, and a finite set at that, since we surely will want to require that algorithms be in principle executable. Now these instructions will be expressions in some language, where a "language" cannot be just an arbitrary set of expressions, but must be accompanied by a semantics, in order that we may know what the instructions require of the computer. It must be admitted that all known languages sutiable for the expression of algorithmic instructions are primitive recursive in their elementary steps, and so algorithms expressed in them are algorithms for recursive functions. But to insist that all possible languages for expressing algorithms must be like the known languages in this respect is to insist upon **CT**. While a language suitable for the presentation of a non-Turing realizable algorithm would have to be unlike known languages, it would have this in common with them: there would be a decision procedure for the set of meaningful expressions of the language. We want algorithms to be recognizably algorithmic. Moreover, decidability of a set would seem to imply the effective enumerability of that set, and so its amenability to Gödel numbering. While this argument cannot be regarded as conclusive, I claim that the first proposition to which acceptance of **RT** requires our assent has been rendered highly plausible by it. Certainly no greater strain is imposed on our intuitions by this proposition than is done by **CT**.

The second proposition which **RT** requires that we accept amounts to the requirement that, for any formal reconstruction **M** of computability that we adopt, we can decide $T_{\mathbf{M}}$. We have just argued that each $M \in \mathbf{M}$ will be described in terms of (and so will correspond to) a set of instructions. These instructions will all have to be of a certain sort (or will have to belong to a decidable class). Accordingly, we ought to be able to decide, for any number, whether that number is the index of an $M \in \mathbf{M}$ or not. In cases where it is not, our decision machine for $T_{\mathbf{M}}$ will simply indicate a negative answer. In cases where it is, we require of our decision machine that it decide whether there is an output in the number of steps which the third argument place of $T_{\mathbf{M}}$ gives, when **M** is presented with the input given in the second argument place. While our formalization of the notion of **R**-ness gives no clue as to what "steps" are like for any particular **M**, I would claim that the intuitive notion of effectiveness presupposes an intuitive notion of a "step", since we conceive an effective procedure, as applied to any particular problem in the class of problems which it solves, as a "step-by-step" operation, or a finite sequence of discrete operations which we may term "steps". One has only to consult any standard introductory text on recursive function theory to see an intuitive motivation given in these terms.

Now, from the index given in the first argument place, we can construct the set of instructions which describes the M. We can then apply those instructions to the input given in the second argument place and count the "steps" (whatever they are like in **M**) up to the number given in the

third argument place. If, after precisely that number of steps, there is an output, then our machine that decides T_M will indicate an affirmative answer, and otherwise a negative one.

I have argued that any class of machines (or algorithms) which we would accept as an adequate reconstruction of the notion of effectiveness must contain a *universal* machine (or algorithm). Now, that there is a universal Turing machine is a result which is shown only with much formal apparatus and hard work, whereas we have made the existence of such a machine for any reconstruction of effectiveness a part of our definition of what is to count as such a reconstruction. One may reasonably ask whether the incorporation of this proposition into our definition is not open to the charge that we have trivialized a highly non-trivial matter. But we have not made the constructive proof of the existence of a universal machine in any particular formal reconstruction a trivial matter. We have merely formalized the intuition that any adequate reconstruction of effectiveness would contain a universal machine. I would claim that if one could not prove the existence of a universal Turing machine, for example, that fact would be a compelling reason to reject Turing computability as an adequate reconstruction of effective computability.

In addition to the philosophical interest I claim for **R**-ness, it seems to me to have some pedagogical usefulness. It is commonly thought useful to instruct beginning students in the method of the standard unsolvability results, without troubling them with all the formal details. In so doing, one makes free use of **CT**, but without really identifying the exact meaning of "recursive" or "Turing computable" or whatever. Our generalization formalizes this pedagogical procedure, and clearly identifies the underlying assumptions associated with it. Perhaps most importantly, our proofs alleviate the excessive informality (or "hand-waving" character) of the standard informal expositions of these results, thereby lending them a bit more security.

REFERENCES

[1] Mostowski, A., *Sentences Undecidable in Formalized Arithmetic*, North-Holland, Amsterdam (1964).

[2] Thomas, W. J., "Doubts about some standard arguments for Church's thesis," in R. J. Bogdan and I. Niiniluoto (eds.), *Logic, Language, and Probability*, D. Reidel, Dordrecht, Holland (1973).

Davidson College
Davidson, North Carolina