[1]

RC-510

Some Theorems About Yamada's Restricted Class Of Recursive Functions

Jerrold Siegel

IBM | RESEARCH

IBM CONFIDENTIAL

Printed in U.S.A.

SOME THEOREMS ABOUT YAMADA'S RESTRICTED CLASS OF RECURSIVE FUNCTIONS

by

Jerrold Siegel

International Business Machines Corporation
Thomas J. Watson Research Center
Yorktown Heights, New York

ABSTRACT: In this paper I discuss a modified Turing machine. This machine has an output symbol augmented to each quadrupole. Two of these symbols are 0 and 1. The machine is set computing on a tape of such nature that the computation does not stop. We consider the sequence of output symbols that arises in this calculation. This sequence is called the sequence of a Yamada pair if the distance between occurrences of either 0 or 1 is bounded. We now consider the subsequence of 0's and 1's. We say that f(a) = b if b - 1 ones or zeros occur before the a th one. It is seen that these functions are all monotonic increasing.

It is shown in this paper that these functions neither include nor are included in the primitive recursive functions that are monotonic increasing, but it is also shown that these functions can all be represented in the form $\mu y(R(x,y)$.

Research Paper
RC-510
August 10, 1961

INTRODUCTION

In "Counting by a Class of Growing Automata" [3] Yamada, motivated by a desire to define the notion of "Real Time Computability", displays a class of modified Turing machines which he calls counters.

A counter is an n-tape Turing machine inside a black box. On top of the box are two lightbulbs, a 1-lightbulb and a 0-lightbulb. The Turing machine and tape configurations are such that if the machine were left to itself it would never halt. A button is pressed and depending on the internal state of the machine and the symbols under the tape heads, one of the bulbs light, the machine then computes for a number of steps (instantaneous descriptions) not greater than a fixed constant and stops. The procedure is repeated. The result is an infinite sequence of 1's and 0's. From this sequence a function f is defined. If P(n) is the sequence, then $f(a) = b$ iff $[P(b) = 1 \wedge [\sum_{x=1}^{b} P(x) = a]$.

Yamada [3] shows this class of functions include the polynomial functions of one variable, the exponential functions of one variable, and many others. He also shows this class closed under many basic operations such as composition. His paper does not show where in the scheme of recursive function theory the "Yamada functions" fall, however. An attempt is made here to fill this gap.

The attempt is a negative one in the sense that it is shown:

(1) Not every Yamada function is primitive recursive, and

(2) Not every monotonic increasing primitive recursive function is a Yamada function.

I do prove as a corollary to the first theorem of the paper, a representation theorem, namely all Yamada's functions can be represented as $\mu y(R(x, y)$ where $R(x, y)$ is primitive recursive. This representation is strictly inclusive by (2), however I hope to follow up this result in a later paper by a more complete representation.

The reader of this paper would do well to avoid the proof of lemma 2 on a first reading. The proof is long and does not aid in the understanding of Sections 2 and 3.

SECTION 1.

In this first section the notions that Yamada considered are formalized. This formalization it is felt, captures the ideas of Yamada, although, other possible conotations can be given to certain sections of his paper.

1-tape Turing machines will only be considered. This is for notational convenience, for although the question of whether or not n-tape Yamada machines can be replaced by 1-tape machines is not answered, the theorems are proved independent of this restriction as can be seen by an examination of the proofs themselves.

Throughout this paper the notation of the theory of Turing machines as presented in Davis [1] is used.

(1) Let q_i denote a state symbol from the set Q of state symbols.

(2) Let S_i be a tape symbol from the set S of tape symbols.

(3) Let R and L be the directions.

We will need another set, $(0, 1, \alpha) = \bar{O}$, of output symbols. The final addition to the notation is the use of "$(q_i, S_i, \_, q_l)$" as the general Turing machine quadruple. (In a particular quadruple, the blank is replaced by R or L or S_k.)

Definition 1

(a) A 5-tuple is said to augment a Turing machine quadruple if the first four entries of the 5-tuple are the entries, in order, of the quadruple, and the 5th is in the set $\bar{O}$.

(b) A set of 5-tuples "g" is called a candidate for a Yamada Turing machine (written CY) if there exists a Turing machine "t" such that each 5-tuple augments one and only one of the quadruples of t and each quadruple of t is augmented by some 5-tuple. We say that the CY augments the Turing machine t. Note that a natural map is defined from the Turing machine quadruples (in particular the first two entries) into the set $\bar{O}$ call this map π_g (considered defined on first two entries).

Definition 2

(a) A tape configuration (TC) is a Turing machine tape with a distinguished starting tape position and a distinguished state as in Davis [1].

(b) Let g be a CY, the set T_g of all TC "t" such that the Turing machine g augments does not halt on t, is called the set of tapes for g.

(c) The set $S_g \subseteq T_g$ such that if S_i is the symbol in the distinguished position and q is the distinguished state $\pi_g(q, S_i) = 0$ or 1 is called the set of Starting Tapes for g.

Definition 3

(a) For each CY g and each $t \in T_g$ a function $\bar{f}_{gt}(n)$ is defined from the non-negative integers into the set $\bar{O}$ as follows: $\bar{f}_{gt}(n) = \pi_g(q_i\ S_j)$. q_i is the state of the Turing machine g augments and S_j the symbol under the reading head after n steps of computation (where a step is a single instantaneous description).

(b) A CY g and a $t \in S_g$ are called a Yamada Pair (YP) if there exists a number K called a Y-number such that for no number n, $\bar{f}_{gt}(n) = \bar{f}_{gt}(n+1) = \cdots = \bar{f}_{gt}(n+K) = \alpha$.

(c) A CY g is called a Yamada Machine if there exists a number K such that for all $t \in S_g (g, t)$ are a Yamada Pair and K is a Y-number for the pair where $S_g \neq \phi$.

Definition 4

(a) Let f(n) be a function into $\bar{O}$ then the Reduced Output Function f of $\bar{f}$ is the function $f(n) = \bar{f}(r)$ where α appears exactly r-n times in the sequence $\bar{f}(0)\ \bar{f}(1) \cdots \bar{f}(r-1)$ and $\bar{f}(r) = 0$ or 1.

(b) Call the reduced output function $f_{gt}(n)$ of $\overline{f}_{gt}(n)$ (as in Definition 3) the Reduced Output Function of the Pair (g, t).

The following lemma is immediate from the definition.

Lemma 1

If (g, t) are a YP then $f_{gt}(n)$ is a total function.

The class of functions discussed by Yamada [3] are generated by the reduced output functions of Yamada Pairs. He points out that the reduced output functions can be generated by Yamada Pairs where the second entries are blank tapes.

I would now like to prove a slightly more general lemma to the effect that:

Lemma 2

If (c, t) are a Yamada Pair then there exists a Yamada machine g,with a blank tape configuration $b \in S_g$ such that $f_{(c,t)}(n) = f_{(g,b)}(n)$ for all n.

Proof

Let t be of the form: $S_1 \cdots S_i q_1 S_{i+1} \cdots S_k$ where everything to the right (left) of and including $S_k(S_1)$ is blank.

The machine g we construct begins by looking at the tape square it is reading; if it is blank it records S_{i+1} and moves to the right. If it is not blank it switches to a state q_σ (to be handled later). The 5-tuples of the early part of the machine g are:

(1) $(q_1, B, S_{i+1}, q_2, \pi_c(q_1, S_{i+1}))$

(2) $(q_1, \_\_, B, q_\sigma, 1)$ where the blank ranges over the set of symbols S_1 to S_k and symbols in 5-tuples of c.

(3) $(q_2, S_{i+1}, R, q_3, \alpha)$

We continue the construction of the machine by duplicating the above procedure for each S_{i+m} the only change being α is the only output symbol of any 5-tuple in the continued construction we have:

(1) $(q_{2m-1}, B, S_{i+m}, q_{2m}, \alpha)$

(2) $(q_{2m-1}, \_\_, B, q_\sigma, \alpha)$

(3) $(q_{2m}, S_{i+m}, R, q_{2m+1}, \alpha)$

When we reach S_k we check for a blank one square further, if there is one, we record a special symbol β in the square, β not in c or on t , if the next square is not blank we again switch to q_σ.

The first part of the machine is completed by repeating the process to the left after returning to the starting position. To the "far" left we place the special symbol γ . Then the machine then returns to the starting position. Note that in each 5-tuple of the machine so far, after the initial pair the symbol α appears in the last position. This first part of the computation takes at most 3(k+2) steps.

Suppose q_δ is the highest state we have used so far in the construction then for each 5-tuple in c beginning with q_1 we include one in the set we are constructing beginning with q_δ replacing the other state q_i appearing in the 5-tuple by $q_{\delta+2+i}$ and changing each output to α . To complete this next part of the construction we add a set of 5-tuples "isomorphic" to c with $q_{\delta+2+i}$ replacing q_i for each state symbol.

We now only have to handle the two symbols β, γ and the state q_σ .

The role of $\beta(\gamma)$ is to note the amount of tape we are sure is blank. If we need more tape we check one square beyond $\beta(\gamma)$ and if it is blank we move $\beta(\gamma)$ to that square and proceed with the calculation using the square from which $\beta(\gamma)$ was erased. We have for each state in the isomorphic copy of c in the machine we are constructing the two sets of 5-tuple: $(q_i, \beta(\gamma), R(L), q_{i+2k}, \alpha)$

$(q_{i+2k}, B, \beta(\gamma), q_{i+2k}, \alpha)$

$(q_{i+2k}, \_\_, B, q_\sigma, \alpha)$ Where the __ is all symbols in c, t not blank.

$(q_{i+2k}, \beta(\gamma), L(R), q_{i+2k+1}, \alpha)$

$(q_{i+2k+1}, \beta(\gamma), B, q_i, \alpha')$ Where k is larger than any state number used.

To complete the machine we just add the set of quadruples $(q_\sigma \_\_, R, q_\sigma, 1)$ where the blank is over all symbol used so far in this construction.

A careful examination of the above construction shows a Y-machine that gives the reduced output function of (c, t) if the tape is effectively blank to c and gives some finite sequence of 1's and 0's followed by an infinite sequence of 1's otherwise (such a sequence is always the reduced output function of some Yamada machine by a simple argument). A bound for this machine is seen to be 3(k+2) + 4d where d is any bound of (c, t).

SECTION 2.

In this section I prove the first two theorems of this paper. I show the existance of a strictly monotonic increasing primitive recursive function that is not a Yamada function and I prove the representation theorem discussed in the introduction. These theorems depend upon a lemma that is central to their proofs. It is this lemma that appears to be the main tool in dealing with Yamada machines that this paper contains.

Definition 5

Let $f_{(c,t)}(n)$ be a reduced output function of a Yamada pair. Let $\bar{h}_{(c,t)}(n) = \sum_{i=0}^{n} f_{(c,t)}(i)$ then the function:

$$h_{(c,t)}(m) = \mu n\ [\bar{h}_{(c,t)}(n-1) = m\ ]\ ; n, m > 0$$

is called the *Yamada Function* of the pair (c, t).

I will not discuss the properties of this function except to note that it is strictly monotonic increasing. A complete description of many constructions of common functions as Yamada Functions can be found in [3].

Lemma 3

There is a recursive enumeration of Gödel Numbers (GN) of CY that do not halt on blank tape, such that each Yamada Function is represented by at least one of the CY.

Proof:

The proof of this theorem is simple and I only give a sketch. The theory of Gödel Numbering of Turing machines is presented in Davis [1] is seen to carry over to the Gödel Numbering of CY. In order to be assured that a CY "g" does not halt on a blank we need only check that all symbol state pairs of $Q_g \times S_g$ (where $Q_g(S_g)$ are all the states (tape symbols) in the 5-tuples of g) appear as leading pairs in other 5-tuples. This is seen to be a recursive procedure, so we may enumerate such machines, but this enumeration is the desired one since if a machine has pairs in $Q_g \times S_g$ that are not 1-2 entries and the machine does not halt, then we know these pairs are not used and a simple addition of 5-tuples will give a complete machine with respect to $Q_g \times S_g$ which acts no differently on blank tape. This and lemma 2 gives the proof since the reduced output functions determine the Yamada Functions.

Lemma 4

There is a primitive recursive function OUT(Z, X) such that if Z is the GN of a CY U, and X is the GN of a tape configuration V, then $\text{OUT}(Z, X) = \pi_U(q_i, S_j)$ where q_i is the distinguished state of V and S_j is the symbol in the distinguished position.

Proof:

Again the proof is a typical argument from the theory of Turing machines and will be left undone as will be the following.

Lemma 5

There is a primitive recursive function $Y(X) = y$ such that if X is the GN of a CY, y is the GN of the Turing machine it augments.

Lemma 6

Let MOVE(Z, X, n) be the function such that if Z is the GN of a Turing machine (called Z with its GN) and X is the GN of a tape configuration, then MOVE(Z, X, n) is the GN of the tape configuration resulting from n steps of computation of Z on X if such a computation exists, otherwise $\text{MOVE}(Z, X, n) = 0$. Then MOVE(Z, X, n) is primitive recursive.

Proof:

Consider the modified form of the Davis T-predicate T'(Z, X, y) which is: y is a sequence of steps in the computation of the Turing machine Z on the tape configuration X. This is primitive recursive [1].

Let X be the GN of the tape configuration $S_1 \cdots S_i \, q_j \, S_{i+1} \cdots S_k$ call the "length" of X equal to k+1. Now after n steps of computation the greatest possible length of the resulting tape configuration is k+n+1 (i. e., a series of steps of the form $S_i \cdots S_{k+1} \; q_j S_k \Rightarrow S_i \cdots S_k q_k B \Rightarrow \cdots \Rightarrow S_i \cdots S_k \; \underbrace{BB \cdots B \cdots B}_{n-1}, \; q_e B$).

The number $N = Z \cdot X$ is greater than the GN of any symbol or state in X or in Z so $[Z, X, k] = 2^N \, 3^N \cdots P_r(k+1)^N$ is greater than X and $\Theta(Z, X, k+n) = \prod_{i=1}^{n+1} (P_r(i))^{[Z, X, k+n]}$ is greater than the Gödel Number of any possible n step computation. It is easy to see that $\Theta(Z, X, k+n)$ is primitive recursive.

We now consider the primitive recursive function:

$$\overline{MOVE}(Z, X, n) = \mu y \, [(T'(Z, X, y) \wedge (P(y) = P(X) + n)) \vee (y=0)]$$

$$y \leq \Theta(Z, X, P(X) + n)$$

where P(X) = length of X. Again this is seen to be primitive recursive.

Now $\overline{MOVE}$ gives the GN of the only possible n step computation. To finish we consider the function $\bar{U}(y)$ = exponent of the highest prime in the construction of y. This is primitive recursive [1] and thus:

$$MOVE(Z, X, n) = \bar{U}(\overline{MOVE}(Z, X, n))$$

is primitive recursive and is our function.

Main Lemma 1

There exists a primitive recursive function k(x, y, m, n) such that if x_0 and y_0 are the GN of CY's and m_0 is any integer, then $k_{x_0 y_0 m_0}(n) = k(x_0, y_0, m_0, n)$ is the reduced output function of some Yamada Pair and if x_0 is the GN of a Yamada Machine and m_0 is a bound for x_0 then $k_{x_0 x_0 m_0}(n)$ is the reduced output function of x_0 acting on blank tape.

Proof:

The proof of this theorem will involve the actual construction of the function k. The function, in brief, takes the machine y and uses it as a "clock". The clock tells, if this time is bounded by m, how long y computes between 1-0 outputs. If the clock really has m as a bound then it is the clock of a Yamada Machine. The function we construct computes using x as long as it matches in time of computation the clock of y. If it ever stops matching the clock it puts out an infinite sequence of 1's from then on. In this case we have an infinite sequence of 1's preceeded by a finite sequence of 1's and 0's. (A Yamada Machine reduced output by a simple argument.)

I now construct the function in a series of subconstructs:

Let b be the GN of a blank tape configuration with distinguished symbol q_1. The first function in the construct is:

$$\mathrm{COM}(m_0, z, x) = \begin{cases} \underset{1 \le m \le m_0}{\mu m} \ [\mathrm{OUT}(z, \ \mathrm{MOVE}(Y(z), \ x, \ m) = 1 \text{ or } 0 \,] \\ m_0 \qquad \text{Otherwise} \end{cases}$$

This function gives the number of steps between the start of computation and the next appearance of 1 or 0 if this number is $\le m_0$, otherwise it gives m_0.

$$\mathrm{MOVE1}(m, z, x, n) = \begin{cases} (n = 0) = x \\ (n > 0) = \mathrm{MOVE}\ [Y(z), \ \mathrm{MOVE1}(m, z, x, n-1), \\ \qquad \mathrm{COM}(m, z, \mathrm{MOVE1}(m, z, x, n-1) \,] \end{cases}$$

This function gives the tape configuration resulting from n-1 movements past occurrences of 1 or 0 to the tape configuration as it appears at the nth occurrence of 1 or 0 (assuming initial output 1 or 0).

$$\mathrm{COM1}(m, z, x, n) = \begin{cases} (n = 0) = 0 \\ (n > 0) = \mathrm{COM}(m, Y(z), \ \mathrm{MOVE1}(m, z, x, n-1) \ ) \end{cases}$$

This function gives the number of steps between the n-1 th occurrence of 1 or 0 and the nth (if this is $\le m$, otherwise m) with the starting output called the 0th.

$$\mathrm{SUM}(m, z, x, n) = \sum_{i=0}^{n} \mathrm{COM1}(m, z, x, i)$$

This function has obvious meaning.

We now define a predicate ONZE(z, x, n); this predicate is true if $\mathrm{OUT}(z, \mathrm{MOVE}(z, x, n)\,) = 1$ or 0, false otherwise.

The function (k) is now expressed as follows:

$$k(x, y, m, n) =$$

$$\begin{cases} [[\bigwedge_{i=0}^{n} \text{ONZE}(x, b, \text{SUM}(m, y, b, i)))] \wedge \neg[\bigvee_{k}^{n} \text{ONZE}(x, b, k)]] \Rightarrow [\text{OUT}(x), \text{MOVE}(m, x, b, n))] \\ \qquad k \leq \text{SUM}(m, y, b, n) \\ \qquad k \neq \text{SUM}(m, y, b, i),\ i \leq n \\ 1 \text{ otherwise} \end{cases}$$

Where b is the GN of the blank tape configuration $\cdot B \cdots B\ q_1\ B \cdots B \cdots$

That this function fits the description can be seen by examination.

With the aid of the lemma, that has just been proved, it will be possible to prove two theorems.

Theorem 1

If Z is the GN of a Yamada Machine and m is a bound for it, then the question "Does $f(a) = c$?" is decidable by a primitive recursive predicate (f is the Yamada Function of the pair (z, b)).

Proof:

Consider the primitive recursive function $\overline{k}(z, z, m, n) = \sum_{i=0}^{n-1} k(z, z, m, i)$ from the definition of the function k and of a Yamada Function we see that $f(a) = c$ is equivalent to $[[\overline{k}(z, z, m, c) = a] \wedge [k(z, z, m, c-1) = 1]]$ which is primitive recursive being the conjunction of two primitive recursive predicates. So theorem 1 is proved.

Corollary

Every Yamada Function f can be written in the form $\mu y(R(x, y))$ where R is a primitive recursive predicate.

Proof:

Let $R(x, y)$ be the primitive recursive predicate $[f(x) = y]$.

Theorem 2

There exists a strictly monotonic increasing primitive recursive function that is not a Yamada function.

Proof:

Let $f(n)$ be the enumeration of lemma 3. Let $K(n)$, $L(n)$ be the p. r. functions that give a 1-1 map of integers into pairs of integers [1]. Let $\overline{k}(z, z, m, n)$ be as in the proof of theorem 1. Consider the primitive recursive predicate:

$$Q(n, a, c) = [[\overline{k}(f(K(n)), f(K(n)), L(n), c) = a] \wedge [k(f(K(n), f(K(n)), L(n), c-1) = 1]]$$

This predicate gives an "enumeration" of the predicates $[f_\sigma(a)=c]$ where f_σ ranges over the entire set of Yamada Functions.

Let $C(n, a, c)$ be the primitive recursive charteristic function of this predicate.

Let $\alpha(n)$ be the primitive recursive function $= 1$ when $n = 0$ and 0 otherwise.

Now consider the primitive recursive function:

$$f(1) = 1 \dot{-} C(1, 1, 1)$$
$$f(n) = f(n-1) + 1 + \alpha(C(n, n, f(n-1) + 1))$$

This function is obviously monotonic increasing. To see that it is different than any Yamada Function f_{n_0} that we choose, we let $C(n_0, a, c)$ be its characteristic function, such an n_0 exists by the enumeration property of $C(n, a, c)$ now $f(n_0) = f(n_0-1) + 1 + \alpha(C(n_0, n_0, f(n_0-1) +1))$ now if $f_{n_0}(n_0) = f(n_0-1) + 1$ then $C(n_0, n_0, f(n_0-1)+1)= 0$ and α of that $= 1$ so $f(n_0) = f(n_0-1) +2 \neq f(n_0-1) + 1 = f_{n_0}(n_0)$ if $f_{n_0}(n_0) \neq f(n_0-1) + 1$ then $f(n_0) = f(n_0-1) +1$ so we are done.

SECTION 3.

I now prove a lemma leading to the third theorem. This will show that the Yamada Functions are not included in the primitive recursive functions.

Main Lemma 2

For any recursive function g there exists a Yamada Function f_g such that $g(x) \leq f_g(x)$ for all x.

Proof: (Outline)

The proof proceeds by a series of modifications on the Turing machine Z_g that computes $g(x)$.

(1) Let Z'_g be the Turing machine that duplicates the input string to the right of itself and then computes on the string to the right, i. e. :

$$\text{S}\underbrace{\text{1111111111}}_{x}\text{S}\ \overset{Z'_g}{\overline{\overset{1}{\Rightarrow}\ \text{S}\underbrace{\text{1111111111}}_{x}\text{S}\underbrace{\text{1111111111}}_{x}\text{S}\ \overset{1}{\Rightarrow}}}\ \text{S}\underbrace{\text{1111111111}}_{x}\text{S}\underbrace{\text{1111111111}}_{g(x)}\text{S}$$

(2) Let Z''_g be the Turing machine that results from tacking on to each "halt" of Z'_g a set of quadruples that erase everything to the right of the second S then moves back to the beginning of the string adds a 1 to the length and transfers to the first state of Z'_g.

The total action of the machine is:

$$\text{S}\underbrace{1111111111}_{x}\text{S} \overset{Z''_g}{\overline{\overset{1}{\Rightarrow} \text{S}\underbrace{1111111111}_{x}\text{S}\underbrace{1111111111}_{g(x)}\text{S} \overset{2}{\Rightarrow}}} \text{S}\underbrace{1111111111}_{(x+1)}\text{S}$$

We now construct the Yamada machine f_g for g. For each quadruple that begins with a pair that would have been a halt in Z'_g. We add a 1 in the fifth position, for every other we add a 0 in the fifth position. This gives a Yamada machine with bound 0. To see that its function f_g is $\geq g$ we note that the only time a 1 appears as output symbol is when the tape is of the form:

$$\text{S}\underbrace{1111111111}_{x}\text{S}\underbrace{1111111111}_{g(x)}\text{S}$$

Now by the construction we see that the total number of 1-0 outputs from "SxSg(x)S"to"S(x+1)Sg(x+1)S" is the total number of steps of computation between. So we see $g(x+1) \dot{-} g(x) \leq$ steps of calculation between SxSg(x)S and S(x+1)S(g(x+1))S $= f_g(x+1) \dot{-} f_g(x)$.

<u>Theorem 3</u>

There exists a Yamada Function that is not primitive recursive.

Proof:

It is proved by Ackerman and stated in Kleene [2]that there exists a recursive function g such that for any primitive recursive function h there exists an n such that $g(n) > h(n)$. Consider $f_g(x)$ if this was primitive recursive, there would be n_0 such that $g(n_0) > f_g(n_0)$ but $f_g(n_0) \geq g(n_0)$ so $f_g(n_0) > f_g(n_0)$ a contradiction.

SECTION 4.

In this section I state results, without proof, that are not in the main line of this paper yet seem to yield important information about the Yamada functions.

The study of Yamada machines may be extended in several directions. The particular direction that I will discuss is the result of changing the requirement that there be a fixed Y-number. If we drop this result completely we have:

Theorem 4

Let $f(n)$ be any recursive function that is strictly monotonic increasing, then there is a CY g and a TC t in T_g such that $f_{(g,t)}(n) = f(n)$ where $f_{(g,t)}(n)$ is the Yamada function of the not necessarily Yamada pair (g,t).

If on the other hand we restrict the bound such that the distance between 1-0 outputs is bounded for the nth output by $f(n)$ where f is primitive recursive then we have:

Theorem 5

Let g be a CY and let t be a T_g with a primitive recursive time found, then the question does $h_{(g,t)}(a) = b$ is decidable by a primitive recursive predicate.

Theorem 6

The class of Yamada functions gotten by allowing the bound to be primitive recursive is exactly the class of monotonic increasing functions that can be represented in the form $\mu(y)\ (R(x,y))$ where R is primitive recursive.

BIBLIOGRAPHY

(1)	M. Davis:	Computability and Unsolvability McGraw-Hill Book Company New York, 1958
(2)	S. Kleene:	Introduction to Metamathematics Van Nostrand New York, 1952
(3)	H. Yamada:	Counting by a Class of Growing Automata A Ph. D Dissertation University of Pennsylvania, 1960

ACKNOWLEDGEMENT

The author would like to thank Dr. J. D. Rutledge for his stimulating commentary on various aspects of this paper's topic. The author also would especially like to thank Dr. C. C. Elgot who suggested the topic of this paper and who carefully read the preliminary draft and found many serious omissions that would have left the paper incomplete.