Notre Dame Journal of Formal Logic
Volume XIX, Number 2, April 1978
NDJFAM

A NOTE ON TURING MACHINE REGULARITY AND PRIMITIVE RECURSION

NICHOLAS J. de LILLO

1 *Introduction* The purpose of this paper is to present an explicit Turing machine **Z** which computes any function which is defined by means of primitive recursion from two given computable functions. The formulation of **Z** uses results of Davis [1] and Mal'cev [3], with the added feature that **Z** yields outputs in a standard form, such outputs usable as inputs in subsequent Turing machines which can be activated after **Z** has completed its computation. Such machines as **Z** are defined as n-regular, for a positive integer n. The course of a computation in **Z** follows along lines suggested by Davis [2], for a similar computation using abstract programs instead of Turing machines.

2 *Preliminary concepts* We will assume a general familiarity with [1], explicitly defining only those concepts which are absolutely necessary for the continuity of this discussion. A Turing machine[1] is any non-empty and finite set of quadruples, any one of which assumes the form (i) $q_i \mathsf{S}_j \mathsf{S}_k q_l$, or (ii) $q_i \mathsf{S}_j \mathsf{R} q_l$, or (iii) $q_i \mathsf{S}_j \mathsf{L} q_l$, where i, j, k, l are positive integers. The symbols q_i, q_l are elements of a finite set Q, called the *internal states* of the machine; the symbols S_j, S_k are elements of the set $A = \{1, B\}$ disjoint from Q and called the alphabet of the machine; the symbols L and R are distinct symbols not in $Q \cup A$. It is understood that no two distinct quadruples of a given Turing machine begin with the same first two symbols. The usual meanings are attached to the quadruples: (i) is the instruction which, when the state of the machine is q_i and the symbol S_j is being scanned, erases S_j and prints S_k in its place, the machine then moving to state q_l; (ii) instructs the machine to move one square to the right and change to state q_l when the machine is in state q_i and scans a square with S_j printed there; (iii) is the instruction similar to (ii), except the machine moves one square to the left.

1. Using the terminology of [1], this paper will deal only with *simple* Turing machines, but these results can easily be generalized to the case of relative computability.

Received December 18, 1974

Let $\mathbf{T}$ be a fixed Turing machine. Then $\theta(\mathbf{T})$ will denote the largest subscript of an internal state symbol appearing in $\mathbf{T}$. Furthermore, for any natural number m, $\mathbf{T}^{(m)}$ will denote the Turing machine obtained from $\mathbf{T}$ by adding m to the value of each subscript of an internal state symbol of $\mathbf{T}$. By an instantaneous description of $\mathbf{T}$ we mean any finite string of symbols from $Q \cup A$ containing exactly one element of Q, and where this element of Q is not permitted to be the rightmost member of the string. In any instantaneous description α, we regard the symbol appearing immediately to the right of the internal state symbol q_i of α as the symbol being scanned by $\mathbf{T}$ when $\mathbf{T}$ is in state q_i. If α, β are instantaneous descriptions of $\mathbf{T}$, we write $\alpha \rightarrow \beta(\mathbf{T})$ (or simply $\alpha \rightarrow \beta$ when $\mathbf{T}$ is understood) to signify that β is the result of an application to α of a single quadruple of $\mathbf{T}$. If, for a given instantaneous description β of $\mathbf{T}$, there is no instantaneous description γ of $\mathbf{T}$ such that $\beta \rightarrow \gamma(\mathbf{T})$, we say that β is final with respect to $\mathbf{T}$. Any finite sequence $\alpha_1, \ldots, \alpha_n$ of instantaneous descriptions of $\mathbf{T}$ such that $\alpha_i \rightarrow \alpha_{i+1}(\mathbf{T})$ for $1 \leq i < n$, and such that α_n is final with respect to $\mathbf{T}$, is called a computation in $\mathbf{T}$ with resultant (or output) α_n. We denote this by $\mathrm{Res}_{\mathbf{T}}(\alpha_1) = \alpha_n$.

The set of natural numbers will be denoted by N, and N^n will denote the set of all n-tuples of elements of N. If $x > 0$, we define 1^x (respectively B^x) to be string of length x of the symbol 1 (respectively B) of A. We also define 1^0 and B^0 to be the empty string. If $n \in N$, we define $\overline{n}$ to be 1^{n+1}, and if $\langle a_1, \ldots, a_n\rangle \in N^n$, we define $\overline{\langle a_1, \ldots, a_n\rangle}$ to be the string $1^{a_1+1}B1^{a_2+1}B \ldots B1^{a_n+1}$. We will consider instantaneous descriptions of $\mathbf{T}$ of the form $q_1\overline{\langle a_1, \ldots, a_n\rangle}$ as usable inputs of $\mathbf{T}$. For any instantaneous description α of $\mathbf{T}$, we define $[\alpha]$ to be the number of occurrences of the symbol 1 in α.

If n is a fixed positive integer, then $\mathbf{T}$ is called n-regular if (1) there is a positive integer p such that

$$\mathrm{Res}_{\mathbf{T}}(q_1\overline{\langle a_1, \ldots, a_n\rangle}) = q_{\theta(\mathbf{T})}\overline{\langle b_1, \ldots, b_p\rangle}$$

whenever $\mathrm{Res}_{\mathbf{T}}(q_1\overline{\langle a_1, \ldots, a_n\rangle})$ is defined, and if (2) $\mathbf{T}$ has no quadruple beginning with $q_{\theta(\mathbf{T})}$.

A function F whose domain is a subset of N^n and having values in N is called partially computable if there is some Turing machine $\mathbf{T}$ such that for any $\langle a_1, \ldots, a_n\rangle$ in the domain of F, it is the case that

$$F(a_1, \ldots, a_n) = [\mathrm{Res}_{\mathbf{T}}(q_1\overline{\langle a_1, \ldots, a_n\rangle})].$$

Thus, for an input of the form $\alpha = q_1\overline{\langle a_1, \ldots, a_n\rangle}$, $\mathbf{T}$ will yield an output for just those n-tuples $\langle a_1, \ldots, a_n\rangle$ which happen to be members of the domain of F; otherwise, $\mathbf{T}$ will hot yield an output for the input α. Furthermore, F is called a total function if its domain is all of N^n, and is called computable if it is partially computable and is total. If F is (partially) computable via the Turing machine $\mathbf{T}$, then we say that $\mathbf{T}$ (partially) computes F.

3 *Main result* We prove the following

Theorem *Let n be a positive integer. If f is a total function of $n+1$*

variables, and is defined by primitive recursion in terms of the computable functions g, h, of n and $n + 2$ variables, respectively, i.e., *if for any* $\langle x_1, \ldots, x_n\rangle \in N^n$ *and* $t \in N$,

$$f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n),$$
$$f(x_1, \ldots, x_n, t + 1) = h(x_1, \ldots, x_n, t, f(x_1, \ldots, x_n, t)),$$

then there is an $(n + 1)$*-regular Turing machine* $\mathbf{Z}$ *which computes* f. *More precisely, for any* $\langle x_1, \ldots, x_n, y\rangle \in N^{n+1}$.

$$\mathrm{Res}_{\mathbf{Z}}(q_1\overline{\langle x_1, \ldots, x_n, y\rangle}) = q_{\theta(\mathbf{Z})}\overline{f(x_1,\ldots,x_n,y)}.$$

Proof: Suppose $\mathbf{G}$ is a Turing machine which computes g, and $\mathbf{H}$ is a Turing machine which computes h. By results of [1], we may assume that $\mathbf{G}$ is n-regular, and that $\mathbf{H}$ is $(n + 2)$-regular. Indeed, using results of [1], there then exists an $(n + 1)$-regular Turing machine $\mathbf{V}_1$ such that, for any $\langle x_1, \ldots, x_n, y\rangle \in N^{n+1}$,

$$\begin{aligned}\mathrm{Res}_{\mathbf{V}_1}(q_1\overline{\langle x_1, \ldots, x_n, y\rangle}) &= q_p\overline{\langle g(x_1, \ldots, x_n), y, x_1, \ldots, x_n\rangle}\\ &= q_p\overline{\langle f(x_1, \ldots, x_n, 0), y, x_1, \ldots, x_n\rangle}.\end{aligned}$$

where $p = \theta(\mathbf{V}_1)$. Similarly, using the fact that $\mathbf{H}$ is $(n + 2)$-regular, there is an $(n + 2)$-regular Turing machine $\overline{\mathbf{H}}$ such that, for all $t, y, x_1, \ldots, x_n \in N$,

$$\begin{aligned}&\mathrm{Res}_{\overline{\mathbf{H}}}(q_1\overline{\langle f(x_1, \ldots, x_n, t), y, x_1, \ldots, x_n\rangle})\\ &= q_{\theta(\overline{\mathbf{H}})}\overline{\langle h(x_1, \ldots, x_n, t, f(x_1, \ldots, x_n, t)), y, x_1, \ldots, x_n\rangle}\\ &= q_{\theta(\overline{\mathbf{H}})}\overline{\langle f(x_1, \ldots, x_n, t + 1), y, x_1, \ldots, x_n\rangle}.\end{aligned}$$

In addition, there is a Turing machine $\overline{\overline{\mathbf{H}}}$ such that, for every $t, y, x_1, \ldots, x_n \in N$,

$$\begin{aligned}&\mathrm{Res}_{\overline{\overline{\mathbf{H}}}}(1B^2q_1\overline{\langle f(x_1, \ldots, x_n, t), y, x_1, \ldots, x_n\rangle})\\ &= 1B^2q_{\theta(\overline{\overline{\mathbf{H}}})}\overline{\langle f(x_1, \ldots, x_n, t + 1), y, x_1, \ldots, x_n\rangle}.\end{aligned}$$

In particular, $\overline{\overline{\mathbf{H}}}$ may be regarded as the Turing machine which first moves left to erase the leftmost 1 of the input $1B^2q_1\overline{\langle f(x_1, \ldots, x_n, t), y, x_1, \ldots, x_n\rangle}$, then moves right until a 1 is found. $\overline{\overline{\mathbf{H}}}$ then performs the identical computation which $\overline{\mathbf{H}}$ performs on $q_1\overline{\langle f(x_1, \ldots, x_n, t), y, x_1, \ldots, x_n\rangle}$. After this last computation has been completed, $\overline{\overline{\mathbf{H}}}$ then moves left to reprint a 1 three squares to the left of the leftmost 1 of the resultant β of the $\overline{\mathbf{H}}$-computation. Finally, $\overline{\overline{\mathbf{H}}}$ moves right until it scans the leftmost 1 of β.

Now set $k = \theta(\overline{\overline{\mathbf{H}}}^{(u+3)})$, where $u = p + 10 + 2n$, and let $\mathbf{W}$ be the following set of quadruples:

$$\begin{array}{ll}
q_p 1\,\mathsf{L}\,q_p & q_{p+9}1Bq_{p+10} \\
q_pB\,\mathsf{L}\,q_{p+1} & q_{p+10}B\,\mathsf{R}\,q_{p+9} \\
q_{p+1}B\,\mathsf{L}\,q_{p+1} & \left.\begin{array}{l} q_{p+7+2i}B\,\mathsf{R}\,q_{p+7+2(i+1)} \\ q_{p+7+2(i+1)}1Bq_{p+7+2(i+1)+1} \\ q_{p+7+2(i+1)+1}B\,\mathsf{R}\,q_{p+7+2(i+1)} \end{array}\right\} \begin{array}{l}\text{for}\\ \text{each } i,\\ 1 \leqslant i \leqslant n\end{array} \\
q_{p+1}1\,\mathsf{R}\,q_{p+2} & \\
q_{p+2}B\,\mathsf{R}\,q_{p+3} & \\
q_{p+3}1Bq_{p+4} & q_{p+8}1\,\mathsf{L}\,q_{u+1} \\
q_{p+4}B\,\mathsf{R}\,q_{p+5} & q_{u+1}B\,\mathsf{L}\,q_{u+2}
\end{array}$$

$$\begin{array}{ll} q_{p+5}1\mathsf{R}\,q_{p+5} & q_{u+2}B1q_{u+3} \\ q_{p+5}B\mathsf{R}\,q_{p+6} & q_{u+3}1\mathsf{L}\,q_{u+3} \\ q_{p+6}1Bq_{p+7} & q_{u+3}B\mathsf{R}\,q_{u+4} \\ q_{p+7}B\mathsf{R}\,q_{p+8} & q_{u-1}BBq_{k+1} \\ q_{p+8}B\mathsf{R}\,q_{p+9} & \end{array}$$

Let $\mathbf{E}$ be the following set of quadruples:

$$\begin{array}{ll} q_{k+1}B\mathsf{L}\,q_{k+1} & q_{k+4}B\mathsf{R}\,q_{k+7} \\ q_{k+1}1\mathsf{L}\,q_{k+2} & q_{k+5}B\mathsf{R}\,q_{k+5} \\ q_{k+2}1\mathsf{L}\,q_{k+2} & q_{k+5}1\mathsf{L}\,q_{k+6} \\ q_{k+2}B\mathsf{L}\,q_{k+3} & q_{k+6}B1q_{k+7} \\ q_{k+3}B\mathsf{L}\,q_{k+4} & q_{k+7}B\mathsf{R}\,q_{k+7} \\ q_{k+4}1Bq_{k+5} & q_{k+7}11q_{k+8}. \end{array}$$

Finally, let $\mathbf{Z} = \mathbf{V}_1 \cup \overline{\overline{\mathbf{H}}}^{(u+3)} \cup \mathbf{W} \cup \mathbf{E} \cup \{q_k 11 q_p\}$. We claim that $\mathbf{Z}$ is a Turing machine which computes f. Note first that $\theta(\mathbf{Z}) = k + 8$, and that there is no quadruple of $\mathbf{Z}$ beginning with $\theta(\mathbf{Z})$. Hence the second property of $(n + 1)$-regularity is satisfied by $\mathbf{Z}$. The first property of $(n + 1)$-regularity will be verified if we can show that for any $\langle x_1, \ldots, x_n y\rangle \in N^{n+1}$,

$$\mathrm{Res}_{\mathbf{Z}}(q_1\overline{\langle x_1, \ldots, x_n, y\rangle}) = q_{\theta(\mathbf{Z})}\overline{f(x_1, \ldots, x_n, y)}.$$

This will be done by tracing a computation in $\mathbf{Z}$ beginning with an input of the form $q_1\overline{\langle x_1, \ldots, x_n, y\rangle}$. First of all,

$$q_1\overline{\langle x_1, \ldots, x_n, y\rangle} \to \ldots \to q_p\overline{\langle f(x_1, \ldots, x_n, 0), y, x_1, \ldots, x_n\rangle}$$

(using $\mathbf{V}_1$).

(i) Suppose that $y = 0$. Then

$$q_p\overline{\langle f(x_1, \ldots, x_n, 0), 0, x_1, \ldots, x_n\rangle} \to \ldots$$
$$\to 1B^2 1^{f(x_1,\ldots x_n\ 0)}B^3B^{x_1+1}B\ \ldots\ BB^{x_n+1}q_{k+1}\,B = \alpha_1, \text{ (using } \mathbf{W}).$$

Applying $\mathbf{E}$ to α_1, either of two possibilities exist as the resultant of a computation in $\mathbf{E}$ beginning with α_1; namely,

$$\mathrm{Res}_{\mathbf{E}}(\alpha_1) = \begin{cases} B^2q_{k+8}f(x_1, \ldots, x_n, 0)B^3B^{x_1+1}B\ \ldots\ BB^{x_n+1}B, \text{ in case} \\ \qquad\qquad f(x_1, \ldots, x_n) > 0, \\ B^3q_{k+8}f(x_1, \ldots, x_n, 0)B^5B^{x_1+1}B\ \ldots\ BB^{x_n+1}B, \text{ if} \\ \qquad\qquad f(x_1, \ldots, x_n) = 0. \end{cases}$$

Each of these resultants is final with respect to $\mathbf{Z}$. Hence, if we disregard initial and terminal blocks of 1's, we get

$$\mathrm{Res}_{\mathbf{Z}}(q_1\overline{\langle x_1, \ldots, x_n, 0\rangle}) = q_{\theta(\mathbf{Z})}\overline{f(x_1, \ldots, x_n, 0)},$$

which is the desired result when $y = 0$. Note that, in this case, the machine $\overline{\overline{\mathbf{H}}}^{(u+3)}$ was never activated. Indeed, $\overline{\overline{\mathbf{H}}}^{(u+3)}$ would not be activated unless the function h were used in evaluating f, which is not the case if $y = 0$.

(ii) Now suppose $y > 0$. Then, after $\mathbf{V}_1$ has completed its computation,

$q_p\overline{\langle f(x_1, \ldots, x_n, 0), y, x_1, \ldots, x_n\rangle} \to \ldots$
$\to 1B^2q_{u+4}\overline{\langle f(x_1, \ldots, x_n, 0), y - 1, x_1, \ldots, x_n\rangle}$ (using **W**),
$\to \ldots \to 1B^2q_k\overline{\langle f(x_1, \ldots, x_n, 1), y - 1, x_1, \ldots, x_n\rangle}$ (using $\overline{\overline{\mathbf{H}}}^{(u+3)}$),
$\to \ldots \to 1B^2q_p\overline{\langle f(x_1, \ldots, x_n, 1), y - 1, x_1, \ldots, x_n\rangle}$ (using q_k11q_p),
$\to \ldots \to 1B^21^{f(x_1,\ldots,x_n,y)}B^3B^{x_1+1}B \ldots BB^{x_n+1}q_{k+1}B$,
iterating the sequence, **W**, $\overline{\overline{\mathbf{H}}}^{(u+3)}$, q_k11q_p, until $y = 0$.

Let $\overline{\beta} = 1B^21^{f(x_1,\ldots,x_n,y)}B^3B^{x_1+1}B \ldots BB^{x_n+1}q_{k+1}B$. Then, using the quadruple $q_{k+1}B\mathsf{L}q_{k+1}$ as many times as is applicable, we get either

$$\overline{\beta} \to q_{k+1}1B^5B^{x_1+1}B \ldots BB^{x_n+1}B = \beta_1,$$

if $f(x_1, \ldots, x_n, y) = 0$, or

$$\overline{\beta} \to 1B^21^{f(x_1,\ldots,x_n,y)-1}q_{k+1}1B^3B^{x_1+1}B \ldots BB^{x_n+1}B = \beta_2,$$

if $f(x_1, \ldots, x_n, y) > 0$.

(a) Suppose $f(x_1, \ldots, x_n, y) = 0$; then, using the remaining quadruples of **E**,

$\beta_1 \to \ldots \to B^3q_{k+8}1B^5B^{x_1+1}B \ldots BB^{x_n+1}B$
$= B^3q_{k+8}\overline{f(x_1, \ldots, x_n, y)}B^5B^{x_1+1}B \ldots BB^{x_n+1}B.$

(b) Suppose $f(x_1, \ldots, x_n, y) > 0$; then, using the remaining quadruples of **E**,

$$\beta_2 \to \ldots \to B^2q_{k+8}\overline{f(x_1, \ldots, x_n, y)}B^3B^{x_1+1}B \ldots BB^{x_n+1}B.$$

The resultants obtained in (a) and (b) above are each final with respect to **Z**. Thus, omitting initial and terminal block's of B's, we get

$$\mathrm{Res}_{\mathbf{Z}}(q_1\overline{\langle x_1, \ldots, x_n, y\rangle}) = q_{\theta(\mathbf{Z})}\overline{f(x_1, \ldots, x_n, y)}$$

whenever $y > 0$.

This completes the proof of (ii), and thus the proof of the Theorem is complete.

4 *Additional notes* In case $y > 0$, the machine **Z** decreases the value of y by 1 whenever **W** is activated. The quadruple q_k11q_p acts as a recycling instruction, demanding a repetition of the sequence **W**, $\overline{\overline{\mathbf{H}}}^{(u+3)}$ as many times as is necessary to obtain $y = 0$. When $y = 0$, the machine **E** acts as an exiting mechanism, yielding an output in the standard form $q_{\theta(\mathbf{Z})}\overline{f(x_1, \ldots, x_n, y)}$.

The leftmost 1 in the instantaneous descriptions α_1, β_1, β_2 plays the role of a marker", in the sense that it prevents an infinite leftward movement of **Z** via the quadruple $q_{k+1}B\mathsf{L}q_{k+1}$, in case $y > 0$ and $f(x_1, \ldots, x_n, y) = 0$. This 1 is then erased by **E** as part of its computation.

Finally, if **Z** is augmented by the single quadruple $q_{k+8}1Bq_{k+9}$, then the resulting machine **Z′** has the property of $(n + 1)$-regularity, and, in addition

$$[\mathrm{Res}_{\mathbf{Z}'}(q_1\overline{\langle x_1, \ldots, x_n, y\rangle})] = f(x_1, \ldots, x_n, y),$$

for every $\langle x_1, \ldots, x_n, y\rangle \in N^{n+1}$.

REFERENCES

[1] Davis, M., *Computability and Unsolvability*, McGraw-Hill Book Company, New York (1958).

[2] Davis, M., *Computability*, Courant Institute of Mathematical Sciences, New York University, New York (1974).

[3] Mal'cev, A. I., *Algorithms and Recursive Functions*, Walters-Nordhoff Publishing, Groningen (1970).

Manhattan College
New York, New York