Notre Dame Journal of Formal Logic
Volume VIII, Number 4, October 1967

PRIMITIVE RECURSIVE COMPUTATIONS

STEPHEN H. McCLEARY

1. *Definition of a computation.** Using the definition of primitive recursive function found in Kleene, p. 219 [1], we shall define a (primitive-recursive) computation, investigate the mechanics of executing such a computation, and derive upper bounds for the value of the function and for the number of steps required for the computation.

Kleene's definition is: "Each of the following equations and systems of equations (I)-(V) defines a number-theoretic function ϕ, when n and m are positive integers, i is an integer such that $1 \leq i \leq n$, q is a natural number, and $\psi, \chi_1, \ldots, \chi_m\chi$ are given number-theoretic functions of the indicated numbers of variables.

(I) $$\phi(x) = x'.$$

(II) $$\phi(x_1, \ldots, x_n) = q.$$

(III) $$\phi(x_1, \ldots, x_n) = x_i.$$

(IV) $$\phi(x_1, \ldots, x_n) = \psi(\chi_1(x_1, \ldots, x_n), \ldots, \chi_m(x_1, \ldots, x_n)).$$

(Va) $$\begin{cases} \phi(0) = q, \\ \phi(y') = \chi(y, \phi(y)). \end{cases}$$

(Vb) $$\begin{cases} \phi(0, x_2, \ldots, x_n) = \psi(x_2, \ldots, x_n), \\ \phi(y', x_2, \ldots, x_n) = \chi(y, \phi(y, x_2, \ldots, x_n), x_2, \ldots, x_n). \end{cases}$$

((Va) constitutes the case of (V) for $n = 1$, and (Vb) for $n > 1$.) A function is *primitive recursive* if it is definable by a series of applications of these five operations of definition."

Modifying this definition to permit zero arguments in (II) so that (Va) and (Vb) can be combined, we proceed in the obvious way to give a recursive definition of *function word*, giving in the process a definition of the *rank* of a function word:

(1) S is a function word of rank 1.
(2) C_m^n is a function word of rank n $(n \geq 0)$.
(3) U_m^n is a function word of rank n $(n \geq 1;\ 1 \leq m \leq n)$.
(4) If A^m is a function word of rank m and if $B_1^n, \ldots, B_m^n$ are function words of rank n, then $\mathbf{S}_m^n A^m B_1^n \ldots B_m^n$ is a function word of rank n $(m \geq 1,\ n \geq 1)$.

*Research supported in part by National Science Foundation grant GP 3993.

Received December 9, 1966

(5) If A^{n-1} is a function word of rank n-1 and if B^{n+1} is a function word of rank n+1, then $\mathbf{R}^n A^{n-1} B^{n+1}$ is a function word of rank n ($n \geq 1$).
(6) No expression is a function word unless required to be by (1), . . . , (5).

A *computation expression* is defined by

(1) A numeral is a computation expression.
(2) If W is a function word of rank n ($n \geq 0$) and if $E_1, \ldots, E_n$ are computation expressions, then $W(E_1, \ldots, E_n)$ is a computation expression.[1]
(3) No expression is a computation expression unless required to be by (1) and (2).

Let N_i be the numeral for the number i. Let E_i be any numeral. Let A's and B's be function words. The *replacement rules* for computation expressions are

(1) $S(N_i)$ can be replaced by N_{i+1}.
(2) $C_m^n(E_1, \ldots, E_n)$ can be replaced by N_m.[2]
(3) $U_i^n(E_1, \ldots, E_n)$ can be replaced by E_i.
(4) $\mathbf{S}_m^n A^m B_1^n \ldots B_m^n(E_1, \ldots, E_n)$ can be replaced by $A^m(B_1^n(E_1, \ldots, E_n), \ldots, (B_m^n(E_1, \ldots, E_n))$.
(5a) $\mathbf{R}^n A^{n-1} B^{n+1}(N_0, E_2, \ldots, E_n)$ can be replaced by $A^{n-1}(E_2, \ldots, E_n)$.
(5b) $\mathbf{R}^n A^{n-1} B^{n+1}(N_j, E_2, \ldots, E_n)$ can be replaced by $B^{n+1}(N_{j-1}, \mathbf{R}^n A^{n-1} B^{n+1}(N_{j-1}, E_2, \ldots, E_n), E_2, \ldots, E_n)$ if $j > 0$.

A *computation* is a sequence $\{C_1, \ldots, C_k\}$ of computation expressions such that C_1 is of the form $W(E_1, \ldots, E_n)$, where W is a function word and $E_1, \ldots, E_n$ are numerals, C_k is a numeral, and C_{j+1} follows from C_j by an application of one of the replacement rules to some subexpression of C_j, $j = 1, \ldots, k$-1.[3] Each computation expression (other than a numeral) must contain at least one replaceable subexpression. (If the entire expression is not replaceable, one of its arguments must be a computation expression which is not a numeral. If this in turn is not replaceable, one of *its* arguments must be a non-numeral, and so on. We finally reach a computation expression which has only numerals as arguments and thus is replaceable.) So that a given computation expression will yield a unique computation, we shall adopt the convention that if the current computation expression contains more than one replaceable subexpression, the one beginning farthest to the left will be replaced. The number of steps in a computation is the number of applications of the replacement rules. To no one's surprise, we shall find that any computation must terminate after finitely many steps.

2. *A description of the computation.* The *length* of a computation expression is the number of letters ($S, C, U, \mathbf{R}$) in it. A computation is completed when an expression of length zero is obtained. An application of any one of the first four replacement rules reduces the length of the expression by precisely one since the rules are applied only when the arguments are numerals and thus contain no letters. An application of (5a) reduces the length by more than one. Now suppose that computation has proceeded until the next step will be the first application of (5b). Let M

denote the computation expression about to be replaced. (M may of course be a subexpression of the current computation expression.) M is $\mathbf{R}^n A^{n-1} B^{n+1}(N_j, E_2, \ldots, E_n)$, where all of the arguments are numerals and $j > 0$. Apply (5b). The expression $M^{(1)}$ which replaces M (and is to be thought of as having "descended" from M) is $B^{n+1}(N_{j-1}, \mathbf{R}^n A^{n-1} B^{n+1}(N_{j-1}, E_2, \ldots, E_n), E_2, \ldots, E_n)$, where the second argument is just M with N_j replaced by N_{j-1}. Since the leftmost replaceable expression was previously M, it is now the second argument of $M^{(1)}$, so the next step is an application of (5a) or (5b) to that argument. Indeed, starting with the replacing of M, we get j consecutive applications of (5b) followed by an application of (5a). Letting $M^{(0)}$ be M and letting $M^{(k+1)}$ be the descendant of $M^{(k)}$, $k = 0, \ldots, j$, we obtain through these $j+1$ steps an expression $M^{(j+1)}$ which is a descendant of M. For $k = 0, \ldots, j-1$, $M^{(k+1)}$ is $M^{(k)}$ with the occurrence of $\mathbf{R}^n A^{n-1} B^{n+1}(N_{j-k}, E_2, \ldots, E_n)$ replaced by $B^{n+1}(N_{j-k-1}, \mathbf{R}^n A^{n-1} B^{n+1}(N_{j-k-1}, E_2, \ldots, E_n), E_2, \ldots, E_n)$. $M^{(j+1)}$ is $M^{(j)}$ with the occurrence of $\mathbf{R}^n A^{n-1} B^{n+1}(0, E_2, \ldots, E_n)$ replaced by $A^{n-1}(E_2, \ldots, E_n)$. We obtain a nesting. For example, let $j = 2$. Then $M^{(j+1)} = M^{(3)} = B^{n+1}(1, B^{n+1}(0, A^{n-1}(E_2, \ldots, E_n), E_2, \ldots, E_n))$. The $\mathbf{R}$ with which M began has disappeared.

(i) *A^{n-1} has at least one less $\mathbf{R}$ than did M; so does each B^{n+1}.*

Since $E_2, \ldots, E_n$ are numerals, the computation next deals with $A^{n-1}(E_2, \ldots, E_n)$ as with the initial expression, leaving the rest of the current expression undisturbed. A^{n-1} may contain further $\mathbf{R}$'s. However, it is clear by induction that $A^{n-1}(E_2, \ldots, E_n)$ will ultimately be reduced to a numeral N (by (i)). We postpone a description of this reduction.

The computation next deals with the rightmost copy of B^{n+1}. Again by induction, $B^{n+1}(0, E, E_2, \ldots, E_n)$ will eventually be reduced to a numeral and again we postpone a description. The computation proceeds leftwards through the remaining copies of B^{n+1} which have descended from M. When the leftmost of these has been reduced to a numeral, M itself has been reduced to a numeral. Each of the reductions whose description was postponed proceeded just as did the reduction of M itself.

The reduction of M to a numeral does not complete the computation. However, the part prior to the first application of (5b) created no copies of any letters and the reduction of M to a numeral did not disturb the rest of the expression. Hence no $\mathbf{R}$ in the initial expression I which had a descendant in M has any descendant still remaining. Furthermore, each of the other $\mathbf{R}$'s in I has precisely one descendant. Hence the reduction of M to a numeral has reduced the number of $\mathbf{R}$'s by the number of $\mathbf{R}$'s in M (which is at least one). Therefore, although further M's may need to be computed, the number of $\mathbf{R}$'s must finally reach zero, after which the computation is easily completed.

3. *Bounds.* We have seen that when a subexpression $\mathbf{R}^n A^{n-1} B^{n+1}(N_j, E_2, \ldots, E_n)$ is encountered, the next $j+1$ steps replace it by an expression containing j copies of B^{n+1} and one copy of A^{n-1}. Thus the size of N_j is crucial. Let H denote at any point during the computation the largest numeral appearing in the current line. We shall investigate how H varies through the computation.

In I, pick any two **R**'s. The one on the right may occur in the A of the one on the left, in the B, or in neither, but not in both. We assume that

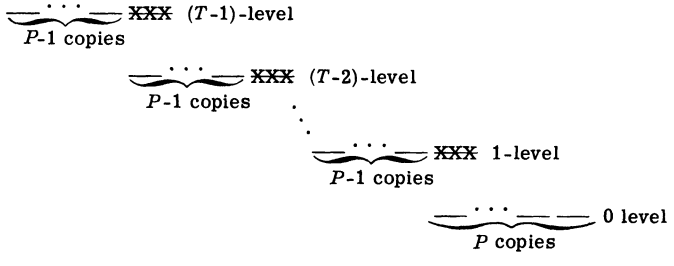
(ii) *For any two* **R**'*s in* I, *the one on the right appears in the* B *of the one on the left.*

This assumption will later be discharged. (We shall see that it results in the highest values of the bounds.) We define several characteristics of I:

Y = largest subscript on any C (= 0 if there are no C's).
Z = largest argument[4].
N = number of occurrences of S with no superscript, i.e., number of occurrences of the successor function.
P = $\max\{Y,Z\}+N$ except that if this is 0, $P = 1$.
T = number of occurrences of **R**.
L = length minus T.

The lack of distinction between numbers and numerals will not cause any confusion. The bounds for the value of the function and the number of steps will be in terms of N,P, and L. The initial size of H is Z. Since we really need only an upper bound for the size of H, we shall assume that initially $H = \max\{Y,Z\}$. This makes it unnecessary to keep track of the introduction of numerals by replacement rule (2). Hence H can be increased only by the elimination of an S according to replacement rule (1). At the first application of (5b), $H \leqslant \max\{Y,Z\} + N \leqslant P$. Assume $H = P$. We process $M = \mathbf{R}^n A^{n-1} B^{n+1}(N_P, E_2, \ldots, E_n)$ getting P copies of B^{n+1} and one of A^{n-1}. Since $P > 0$, we do get copies. By (ii), A^{n-1} contains no **R**'s. We still have $H = P$ after $A^{n-1}(E_2, \ldots, E_n)$ has been replaced by a numeral because each S in A^{n-1} has as ancestor an S none of whose descendants has yet been eliminated; and the ancestors are included in the value of N. Assuming the worst, we suppose that each copy of B^{n+1} is a copy of all of I except that it does not contain the **R** which has just been eliminated. Each of the P copies contains precisely T-1 **R**'s and N S's. The rightmost of the P copies is now "spread out" into P more copies with another **R** eliminated, leaving T-2 **R**'s. The A is computed. Still $H = P$. This process is repeated until finally P copies are obtained with no **R**'s. There are now P-1 nested copies with T-1 **R**'s each, nested within which are P-1 nested copies with T-2 **R**'s each, . . . , nested within which are P-1 nested copies with one **R** each, nested within which are P nested copies with no **R**'s. Still $H = P$. We call the set of copies with k **R**'s the *k-level*, $k = 0, \ldots, T$-1. The situation is suggested by

Figure I



The 0-level is computed. In each of the P copies at the 0-level are N S's, so H increases by PN to give $H = P + PN = P(N+1)$. Any S's encountered after reaching the 1-level, but before the next application of (5b) need not be considered because N S's were allowed in each copy on the 0-level. The rightmost copy on the 1-level is spread out into $P(N+1)$ copies which constitute a new 0-level. This 0-level is computed. Now $H = P(N+1) + (P(N+1))N = P(N+1)^2$. This process is repeated until the 2-level is reached. By induction we see that now $H = P(N+1)^P$. The last copy on the 2-level is spread out into $P(N+1)^P$ copies which constitute a new 1-level. By the result just obtained for 1-levels, computation of the new 1-level yields

$$H = [P(N+1)^P]\ (N+1)^{[P(N+1)^P]}\ .$$

Thus motivated, we define

$$f_0(P,N) = P,$$
$$f_1(P,N) = P(N+1),$$
$$h_{2,1}(P,N) = P(N+1)^P,$$
$$h_{j,k}(P,N) = h_{j,1}(h_{j,k-1}(P,N),N), \quad j \geqslant 2,\ k \geqslant 2,$$
$$h_{j,1}(P,N) = h_{j-1,P}(P,N), \quad j \geqslant 3,$$
$$f_j(P,N) = h_{j,1}(P,N), \quad j \geqslant 2.$$

We have seen that when the k-level is reached, $H = f_k(P,N)$, $k = 0,1,2$. Proceeding by induction, we assume that when the k-level is reached $H = f_k(P,N) = h_{k,1}(P,N)$, $k \geqslant 2$. Thus reducing the P^{th} (rightmost) copy on the k-level to a numeral, which is what has been done when the k-level has been reached, has raised H from P to $h_{k,1}(P,N)$. By the same argument, computing one more copy on the k-level raises $h_{k,1}$ to $h_{k,1}(h_{k,1}(P,N),N) = h_{k,2}(P,N)$. By a subinduction it is clear that upon reaching the $(k+1)$-level, i.e., after computing all copies on the k-level, $H = h_{k,P}(P,N) = h_{k+1,1}(P,N) = f_{k+1}(P,N)$.

When we reach the T-level, $H = f_T(P,N)$ and we have finished computing M. By assumption (ii), no **R**'s remain in the current expression. Each remaining S was assumed to be in each of the copies; hence its effect on H can be neglected. We have proved

Theorem 1: *The value of a primitive recursive function does not exceed* $f_T(P,N)$.

Assume $N > 0$. To find a bound for the number of steps in the computation, we first find how many copies of I have been made. Assume for the moment that only copies on the various 0-levels need be counted. Let H_1 be the value of H as we begin to spread out a 0-level and H_2 the value as we begin to spread out the next 0-level. Then $H_2 = H_1 + H_1N = H_1(N+1)$. Since $N > 0$, $H_1 \leqslant \frac{1}{2}H_2$. At the end of the computation $H = f_T(P,N)$. At the last creation of a 0-level, $H \leqslant \frac{1}{2} f_T(P,N)$. We see that the total number of copies on the various 0-levels does not exceed $f_T(P,N)(\frac{1}{2} + \frac{1}{4} + \ldots) = f_T(P,N)$. Since each copy on the 0-level contains no **R**'s, we have (under the assumption that only copies on the 0-levels need be counted)

Theorem 2: *The number of steps in a primitive recursive computation does not exceed* $f_T(P,N) \cdot L$ *(provided* $N > 0$*).*

If $N = 0$, we may obtain a bound by assuming $N = 1$. A sharper bound is obtainable by leaving N at 0, but its derivation is too tedious to be worth the while. Also, it is clear that if $T = 0$, the bound of $P \cdot L$ given by the theorem can be replaced by L.

It remains to show that only copies on the 0-levels need be counted in obtaining the bound of theorem 2. For $T = 0$, there is nothing to show. Let us look at $T = 1$. M is of the form $\mathbf{R}^n A^{n-1} B^{n+1}(E_1, \ldots, E_n)$ and contains no **R**'s other than the one exhibited. Any steps taken before applying (5b) ((5b) rather than (5a) because $P > 0$) to M will be assumed to occur during the computation of each copy on the 0-level (of which there is at least one because $P > 0$) and thus need not be counted. The steps used to reduce $A^{n-1}(E_2, \ldots, E_n)$ to a numeral are executed before the computation of the 0-level begins, but need not be counted since A^{n-1} is assumed to be included in each copy on the 0-level. The 0-level actually contains (no more than) P copies, but the theorem allows for $P(N+1) \geq 2P$ copies, so we have allowed for P extra copies. This takes care of the steps used to spread out the 0-level. Applying a similar argument at higher levels, it is clear that also for $T > 1$, only copies on the 0-level need be counted.

We shall now justify assumption (ii). The subexpression $\mathbf{R}^n A^{n-1} B^{n+1}(H, E_2, \ldots, E_n)$ is converted into H copies of $B^{n+1}(H > 0)$ and one of A^{n-1}. An **R** contained in B^{n+1} is copied H times; an **R** in A^{n-1}, but once. Since more occurrences of **R** cause a higher function value and more steps in the computation, it may be assumed that no **R** in I lies in the A of another **R**. It may likewise be assumed that for any two **R**'s in I, it is not the case that the one on the right lies in neither the A nor the B of the one on the left. Hence we may assume (ii).

Tabulating the results for small T and recalling that the number of steps does not exceed L if $T = 0$, we get the bounds[5]

Figure II

T	Function Value	Steps
0	P	L
1	$P(N+1)$	$P(N+1) \cdot L$
2	$P(N+1)^P$	$P(N+1)^P \cdot L$

We conclude with an example. An expression for adding 3 and 2 is $\mathbf{S}_2^2\mathbf{R}^2 U_1^1 \mathbf{S}_1^3 S U_2^3 U_2^2 U_1^2(3,2)$. $T = 1$, $N = 1$, $P = 4$, $L = 7$. The function value cannot exceed 8; the actual value is 5. The number of steps cannot exceed 56; the computation actually requires 14 steps.

NOTES

1. We have $n = 0$ only for the words C_m^0. Affixing zero arguments to C_m^0 to get a computation expression is understood to yield simple C_m^0.
2. We interpret replacement rule (2) to mean that C_m^0 can be replaced by N_m.
3. The replacement rules and the definition of computation were developed by Dr. Alan Robinson of Rice University.
4. There will be at least one argument unless I is precisely C_m^0. Since it turns out that the bounds hold for this case, it will be assumed that there are arguments. Hence Z is defined.
5. We recall that if $N = 0$, N must be set to 1 to use the table.

REFERENCE

[1] Kleene, S. C., *Introduction to Metamathematics*, D. Van Nostrand Company, Inc., Princeton, N. J., 1950.

University of Wisconsin
Madison, Wisconsin