DEGREE OF DIFFICULTY OF COMPUTING A FUNCTION AND A PARTIAL ORDERING OF RECURSIVE SETS

Michael O. Rabin

HEBREW UNIVERSITY

Technical Report No. 2

THIS REPORT WAS PREPARED FOR THE UNITED STATES OFFICE OF NAVAL RESEARCH INFORMATION SYSTEMS BRANCH, UNDER CONTRACT

No. 62558-2214

Jerusalem, Israel

25 April 1960

DEGREE OF DIFFICULTY OF COMPUTING A FUNCTION AND A PARTIAL ORDERING OF RECURSIVE SETS

Michael O. Rabin

HEBREW UNIVERSITY

Technical Report No. 2

THIS REPORT WAS PREPARED FOR THE UNITED STATES OFFICE OF NAVAL RESEARCH INFORMATION SYSTEMS BRANCH, UNDER CONTRACT

No. 62558-2214

Jerusalem, Israel

25 April 1960

S U M M A R Y

We attempt to measure the amount of work inherent in the task of computing a given computable (recursive) function. A notion of degree of difficulty of computing a function is introduced and studied. The notion is invariant in the sense that it is independent of the idealized computers (Turing Machines) used for computing the functions in question. Applications are made to the classification of solvable decision problems (recursive sets) according to relative difficulty.

DEGREE OF DIFFICULTY OF COMPUTING A FUNCTION AND A PARTIAL ORDERING OF RECURSIVE SETS

Michael O. Rabin

INTRODUCTION

The operation called "middle-squaring" is sometimes used for the generation of random digits. Let p be an integer having k digits in decimal notation, $m(p)$ is defined to be the k digit integer forming the middle section of p^2 (which is a $2k$-digit integer); in case the middle section of p^2 starts with the digit 0, $m(p)$ is the k digit integer obtained from the middle section by changing the first digit into a 1. Let $m^{-1}(n)$ be the smallest integer p for which $m(p)=n$ and 0 if no such p exists. The only known method for computing $m^{-1}(n)$, where n is a k digit integer, is to calculate one by one all the 9.10^{k-1} values $m(10^{k-1})$, $m(10^{k-1}+1),\ldots,m(10^k-1)$, and thereby find the smallest p (if it exists) for which $m(p)=n$. If we compare the function m^{-1} with the function $d(n)=2n$ we thus feel that m^{-1} is considerably more difficult to compute than d.

The question at once arises how could one assign a precise meaning to the notion "f is more difficult to compute than g"? This question breaks naturally into two parts. First, for a given function f and a given computation of a functio value $f(n)$, we want to express, prefer-

ably by means of a number, the amount of difficulty involved in that computation. The computation of the value $y=f(n)$ may be considered as a proof, in some formal system, of the formula $f(n)=y$; thus what is needed is a measure on proofs. One could, for example, take the length of a proof as a measure of its difficulty. Generally, however, the measure on proofs one would want to use would depend on the methods or computers he has for constructing proofs. It is conceivable that the difficulty involved in the construction of a proof by a certain computer (expressed, say, by the time required by the computer for the construction) depends not on the length of that proof but on some other significant feature of the proof. We therefore introduce, in Section 1, the notion of a measure on proofs axiomatically and do not use any specific measure in the subsequent development. The results therefore apply to any measure on proofs satisfying the axioms. Still, a careful study of measures on proofs would be of great interest on its own rights and also of importance for the theory of computers and automata.

Having agreed upon some measure on proofs, how should the notion "f is more difficult to compute than g" $(g \prec f)$ be defined? The notion must clearly depend on the overall behaviour of the functions. Thus $g \prec f$ if in general it is more difficult to compute $f(n)$ than to compute $g(n)$. Returning to the example of d and m^{-1}, we realize that saying that $d \prec m^{-1}$ because of the fact that with a certain particular method for computing m^{-1} it is harder to compute m^{-1} than to compute d, is not justified. It is possible (though

improbable) that there exists some algorithm for computing $m^{-1}(n)$ which is, in general, simpler than the computation of $d(n)=2n$. To get an adequate invariant definition for $g \prec f$ we must therefore consider all possible algorithms for computing f. The definition of $g \prec f$ as well as the main theorems concerning this relation are contained in Section 2.

The most general notion of an algorithm existing in the literature is that arising from Post Systems [3, 152-169]. A Post system L consists of a finite set of symbols $a_1,\ldots,a_n$, called the alphabet of L; a finite set of words $A_1,\ldots,A_m$ on this alphabet, called the axioms of L; and a finite set of production rules of the form

$$\begin{array}{l} \sigma_{11} P_{11} \sigma_{12} P_{12} \cdots P_{1k_1} \sigma_{1k_1} \\ \cdot \\ \cdot \\ \cdot \\ \sigma_{r1} P_{r1} \sigma_{r2} P_{r2} \cdots P_{rk_r} \sigma_{rk_r} \end{array} ,$$

produces $\tau_1 P_1 \tau_2 P_2 \cdots \tau_s P_s \tau_{s+1}$, where the σ_{ij} and τ_i are fixed words, the P_{ij} are variable words and $P_1,\ldots,P_s$ are a subset of the words P_{ij}. The notions of a proof of L and theorem of L are now defined in the usual way. All our results will be stated in terms of Post systems but they remain true for any other notion of algorithm.

By the decision problem of a set S of integers we mean the problem of deciding for each integer i whether $i \in S$. This terminology conforms to

that of Post's article [2] but differs from the terminology of Tarski [4]. The decision problem of S is called solvable if S is recursive. The notions and results of Section 2 now enable us to order solvable decision problems according to their relative degree of difficulty (Section 3). Stronger results are obtained for the case of primitive recursive sets. Whereas in Post's ordering by means of recursive reducibility all recursive sets are lumped into one class, a whole spectrum of classes of recursive, and even primitive recursive, sets is obtained under the ordering defined here. Further research will be needed for determining the exact form of the partially ordered system obtained by our ordering relation.

In Section 4 the results about ordering of solvable decision problems according to degree of difficulty are applied to proving that certain Post systems are undecidable.

1. MEASURES ON PROOFS AND LENGTH OF COMPUTATION FUNCTION.

Definition 1. A function $m(L,P)$ defined for pairs L,P, where L is a Post system and P is a proof of L, and assuming integral values, will be called a measure on proofs if the following conditions are satisfied. (a) m is primitive recursive. (b) Given a system L and a number n there is only a finite number $\nu(L,n)$ of proofs P of L such that $m(L,P) \leqslant n$. (c) The function $\nu(L,n)$ is

primitive recursive.

Remark 1. Strictly speaking the function m depends not only on the proof P but also on the Post system L in which P is a proof. The same string P of words may be a proof with respect to two different systems L_1 and L_2 and it is possible that $m(L_1,P) \neq m(L_2,P)$. Still, when there is no danger of confusion we shall drop the reference to L and simply write m(P).

An immediate consequence of the above requirements is the following

Lemma. If m is a measure then there exists an effective primitive recursive procedure whereby when given a system L and a number n it is possible to find all proofs $P_1,\ldots,P_k$ $(k=\varphi(L,n))$ of L for which $m(P_i) \leq n$.

Example 1. The length l(P) of a proof (i.e. the number of words in the sequence P) is a measure on proofs. It is not hard to verify that condition (a)-(c) are satisfied by this function.

Example 2. Taking the length as a measure on proofs is motivated by the model of a procedure for constructing proofs in which the various steps are taken in linear order one at a time. But we might conceive of a procedure or a computer which can perform several operations simultaneously. A computer may possess facilities for drawing in one step several immediate consequences from a given set of words. In actual cases there will exist an a-priori bound to the number of operations which a given computer may perform simultaneously. For each

bound N we could define a measure m_N on proofs which is motivated by the model of computers capable of performing N operations simultaneously. From our point of view it is, however, more interesting to model a measure on a more extreme assumption, namely that at each stage in the construction of a proof we draw <u>in one step</u> all the necessary immediate consequences from the words already derived. This motivates the following definition.

Let $P = \sigma_1, \ldots, \sigma_n$ be a proof of the system L. Define inductively

$$S_1 = \{ \sigma_i \mid \sigma_i \in A \}$$

(where A is the set of axioms of L), and in general

$$S_k = S_{k-1} \cup \{ \sigma_i \mid \sigma_i \text{ immediate consequence of words in } S_{k-1} \}$$

The number $d(L,P)$, to be called the <u>depth</u> of P, is now defined as the smallest d for which $S_d = P$.

Again it is possible to prove that $d(L,P)$ is a measure on proofs. Clearly $d(L,P) \leqslant l(P)$ for all proofs P. The measure d is, in fact, a lower bound of all the previously mentioned measures m_N; thus $d(L,P) \leqslant m_N(L,P)$ for all measures m_N.

In order to fix notations we shall henceforth assume that all Post systems under consideration contain the stroke | and equality

sign = in their alphabet. The word $||\ \ldots\ |$ consisting of $n+1$ strokes will be referred to as the numeral n and denoted by $\overline{n}$.

Definition 2. Let L be a Post system and ω a word on the alphabet of L. The pair (L,ω) is said to compute the function f if for all integers n,m

$$\vdash_L \omega\overline{n}=\overline{m} \longleftrightarrow f(n)=m.$$

We shall usually drop the reference to ω and simply say that L computes f. A function f is computable (recursive) if and only if there exists a system L computing it.

The computation of a function f by a given system (L,ω) presents a certain amount of work which we would like to measure. This is achieved by the function introduced in the following

Definition 3. Let (L,ω) compute f and let m be a measure on proofs. The length of computation function $F_L(n)$ is defined by

$$F_L(n) = \min_{P\in C} m(L,P)\ , \quad C = \{P \mid P \text{ proof of } \omega\overline{n}=\overline{m} \text{ in } L\}$$

For a given function f and system (L,ω) computing it, the length of computation function obviously depends both on ω and the measure m used. In any given discussion, however, both ω and m are kept fixed and we simply write $F_L(n)$.

Remark 2. It is easy to verify that for a given system L which computes f, the function $F_L(n)$ is coputable. If f is primitive recursive then there exists a system (L, ω) computing f such that $F_L(n)$ is primitive recursive. Namely, let E be the system of defining equations for f constructed according to Kleene's proof that every primitive recursive function is general recursive [1, pp. 262-270]. Let L be a Post system obtained by a straightforward translation of the formal system consisting of E and Kleene's rules for deducing equations, into the framework of Post systems (we omit the simple details of this prpcess). For this L the function F_L is primitive recursive.

2. THE RELATION $g \prec f$ AND ITS PROPERTIES.

The task of comparing the respective degrees of difficulty involved in computing two functions is rather delicate. Given two computable functions f and g we may chose Post systems L and M which compute the functions f and g respectively and compare the length of cmputation functions $F_L(n)$ and $G_m(n)$. But this is not adequate because using other Post systems for computing f and g may change the outcome of this comparison. The natural way to get an invariant notion depending only on the functions is as follows.

Definition 4. A computable function f is said to be more difficult

to compute than a computable function g (in symbols: $g \prec f$) if there exists a system M computing g such that for <u>every</u> L computing f there exists an integer n_o for which

(1) $n_o < n \longrightarrow F_M(n) \prec F_L(n)$.

<u>Remark</u> 3. The introduction of n_o in the above definition is unavoidable. For it is always possible to "cheat" and for any given k define a system (L, ω) computing f such that for all $n \leqslant k$ the word $\omega\bar{n}=\overline{f(n)}$ is included among the axioms of L. For any natural measure m on proofs (e.g. the measures of 1 and d of Examples 1-2) we would then have $F_L(n)=1$. Thus $F_m(n) \prec F_L(n)$ can not be expected to hold for all n; at best it can hold from a certain integer n_o and on.

<u>Theorem</u> 1. For all computable functions f, $f \not\prec f$.

The proof is immediate and will be omitted.

<u>Theorem</u> 2. If $g \prec f$ and $f \prec h$ then $g \prec h$.

<u>Proof</u>. For appropriate systems M and N computing g and f respectively we have

(2) $G_M(n) < F_L(n)$ for $n(L) \leqslant n$,

(3) $F_N(n) < H_K(n)$ for $n'(K) \leqslant n$,

where L and K are arbitrary systems computing f and h. Taking $L=N$ we have

(4) $G_M(n) < H_K(n)$ for $\max(n(N), n'(K)) \leqslant n$.

Corollary. If $g \prec f$ then $f \not\prec g$.

This follows at once from the previous two theorems.

The relation $\prec$ expressing the relative difficulty of computation has, in view of Theorems 1-2 and the Corollary, all the properties of a partial ordering. It is not obvious from the start that this partial ordering is non-trivial, i.e. that there are at all functions f and g which are comparable. The following two theorems supply information on this point but actually accomplish much more.

Theorem 3. Let h be a computable function. There exists a computable function f assuming only the values 0,1 and primitive recursive in h such that for every system L which computes f, $h(n) < F_L(n)$ for almost all integers n.

Proof. The proof will proceed in two steps. First a weaker result will be established. From this result the desired result will then directly follow.

Let (L, ω) be a pair consisting of the Post system described in the Introduction and a word ω on the alphabet of L. Define the weight $W(L, \omega)$ of the pair by ($l(\sigma)$ denotes the length of the word σ)

$$W(L, \omega) = n + l(\omega) + \sum l(A_i) + \sum l(\sigma_{ij}) + \sum l(P_{kr}).$$

We shall construct a function f assuming only the values 0,1

and primitive recursive in h such that for every system (L,ω) which computes f, $h(n) < W(L,\omega)+F_L(n)$ for almost all n.

For each $0 < k$ let us enumerate all pairs (L,ω) for which $W(L,\omega)=k$ and order them effectively in a finite sequence $(L_{k,1},\omega_{k,1}),\ldots,(L_{k,r(k)},\omega_{k,r(k)})$. This can be done by a primitive recursive procedure which is uniform in k. The function $r(k)$ giving the number of pairs of weight k is primitive recursive. Now we arrange all these pairs $(L_{k,j},\omega_{k,j})$, where $j \leqslant r(k)$, into a single sequence (L_i,ω_i), $i=1,2,\ldots$, by ordering the pairs (k,j) of indices lexicographically. Clearly we obtain an effective, in fact even primitive recursive, enumeration of all pairs (L,ω). This enumeration satisfies

$$W(L_i,\omega_i) \leqslant k \longleftrightarrow i \leqslant r(1) + \ldots + r(k).$$

We now define by induction on n both the function $f(n)$ and an auxiliary sequence of finite sets $I_0, I_1,\ldots,I_n,\ldots$.
Define $I_0=\emptyset$. For every $1 \leqslant n$ enumerate all pairs (L_i,ω_i) for which $i \notin I_{n-1}$ and $W(L_i,\omega_i) \leqslant h(n)$. For each such (L_i,ω_i) enumerate all the proofs P_{ij} of L_i satisfying

(5) $\quad W(L_i,\omega_i) + m(P_{ij}) \leqslant h(n)$.

Let $i=i(n)$ be the smallest integer satisfying the above conditions and the further condition that one of the proofs $P_{i(n),j}$ of $L_{i(n)}$ is a proof of one of the formulas

(6) $\omega_i \overline{n} = \overline{0}$, $\omega_i \overline{n} = \overline{1}$.

If just one of these formulas has a proof $P_{i(n),j}$, define $f(n)$ so that the provable formula is $\omega_i \overline{n} = \overline{1-f(n)}$. If both formulas have proofs among the $P_{i(n),j}$, define $f(n) = 1$. In either case define $I_n = I_{n-1} \cup \{ i(n) \}$.

If for every i satisfying the above conditions there does not exist a proof P_{i_j} satisfying (5) which is a proof of one of the formulas (6), define $f(n) = 1$ and $I_n = I_{n-1}$.

The function f is clearly well defined for all n, assumes only the values 0,1, and is primitive recursive in h.

Assume now that (L_i, ω_i) computes f and that for infinitely many integers p

(7) $W(L_i, \omega_i) + F_{L_i}(p) \leq h(p)$

holds. It is impossible that $i \in I_n$ for some n, for this implies that (L_i, ω_i) does not compute f. Since $I_0 \subseteq I_1 \subseteq I_2 \subseteq \ldots$, there exists an integer s such that for $s < q$ the set I_q contains all the integers $j < i$ satisfying $j \in \bigcup_n I_n$. Chose an integer p satisfying (7) and $s < p$. The integer i satisfies $i \notin I_{p-1}$ and in virtue of (7) there exists in L_i a proof P_{ik} of a formula of the form $\omega_i \overline{p} = \overline{e}$ where e is 0 or 1 (namely $e = f(n)$).

No integer $j < i$ has these two properties, for otherwise we would have for the smallest integer $j_0 < i$ with these properties, $j_0 \notin I_{p-1}$ and hence $j_0 \in I_p$, contrary to $s \leq p-1$ and the definition of s. Thus we get $i \in I_p$, a contradiciton.

To finish the proof of Theorem 3, apply the preliminary result to the function $h(n) + n$. There exists a function f assuming only the values 0,1 and primitive recursive in $h(n)+n$ (hence primitive recursive in h) such that for every (L,ω) computing f there exists an integer n_0

$$n_0 < n \longrightarrow h(n) + n < W(L,\omega) + F_L(n).$$

Thus

$$\max (W(L,\omega), n_0) < n \longrightarrow h(n) < F_L(n),$$

this completes the proof.

Theorem 4. Let g be a computable function. There exists a computable function f assuming only the values 0,1 which is more difficult to compute than g.

Proof. Let M be a system computing g. The function $G_M(n)$ is computable. Apply Theorem 3 to $h(n) = G_M(n)$.

Remark 4. The requirement that f be a function assuming only the values 0,1 is an important part of the claims of Theorems 3 and 4. If a function $f(n)$ grows very rapidly with n then for certain measures

on proofs the function $F_L(n)$ may grow rapidly with n for every system (L, ω) computing f. The words $\omega\overline{n} = \overline{f(n)}$ simply become long and their proofs must involve a large number of steps just to build these words up from the axioms of L. For the measure l on proofs this would make $F_L(n)$ large for large values of n. If, on the other hand, a function f assuming only the values $0,1$ is more difficult to compute than a function g, then we feel that f is really more complex than g, not merely much larger. The fact that we prove Theorems 3-4 with two valued functions is also essential for the following study of the relative difficulty of solvable decision problems

3. ORDERING ACCORDING TO RELATIVE DIFFICULTY OF SOLVABLE DECISION PROBLEMS

Definition 4. A recursive set R will be said to posses a more difficult decision problem than a recursive set S, in symbols: $S \prec R$, if for the respective characteristic functions f_R and f_S, $f_S \prec f_R$ holds.

Theorem 5. For every recursive set S there exists a recursive set R satisfying $S \prec R$.

Proof. Let f_S be the characteristic function of S. By Theorem 4 there exists a function f assuming only the values $0,1$ such that $f_S \prec f$. Define $R = \{ n \mid f(n)=1 \}$. The set R is recursive and has the desired properties.

A stronger result can be obtained for the case that S is a primitive recursive set.

Theorem 6. For every primitive recursive set S there exists a primitive recursive set R satisfying $S \prec R$.

Proof. The characteristic function f_S of S is primitive recursive. According to Remark 2 there exist a system L computing f_S such that the length of the computation function F_L of f_S is primitive recursive. By Theorem 3 there exists a function f assuming only the values $0,1$ and primitive recursive in F_L, satisfying $F_L \prec f$. The set $R = \{ n \mid f(n) \}$ is clearly primitive recursive and satisfies $S \prec R$.

Example 3. Taking S to be the set P of all primes, there exists a primitive recursive set R such that its decision problem is more difficult than that of P.

4. AN APPLICATION TO UNDECIDABILITY OF POST SYSTEMS

Theorem 3 may be generalized in the following way.

Theorem 7. Let $p(n,m)$ be a recursive function. There exists a computable function f assuming only the values $0,1$ such that for every system L computing it and for every fixed integer m_0

$$p(n,m_0) < F_L(n) \quad \text{for almost all integers } n. \tag{8}$$

Proof. Define

$$h(n)= \sum_{k,m \leq n} p(k,m).$$

The function h is recursive. For every m_0 we have that $p(n,m_0) \leqslant h(n)$ for all $n \geqslant m_0$. Now apply Theorem 3 to h.

In [4, p. 49] Mostowski Robinson and Tarski generalize Gödel's argument to prove that every formal theory in which all recursive functions are representable is essentially undecidable. It turns out that an even stronger result holds for arbitrary Post systems. Thus the above mentioned theorem does not really depend on the availability of logical machinery in the formal theories with respect to which it is formulated.

Theorem 8. Let L be a Post system such that for every recursive set S there exists a word ω_S on the alphabet of L such that (L,ω_S) computes the characteristic function of S. The system F is undecidable.

Proof. Assume to the contrary that F is decidable. The function $p(\mathfrak{G})$ from the set of all words on the alphabet of L into integers defined by $p(\mathfrak{G})=k$ if $\vdash_L \mathfrak{G}$ and

(9) $k=\min m\ (L,P)$ for all proof P of $\mathfrak{G}$,

and $p(\mathfrak{G})=0$ if $\mathfrak{G}$ is not a theorem of L, is computable.

Let g be a fixed Gödel numbering of all words on the alphabet of L assuming all integers as values. Define a function $p(n,m)$ by

(10) $g(\omega)=m \longrightarrow p(n,m)=p(\omega\ \overline{n}=\overline{1})+p(\omega\ \overline{n}=\overline{0})$.

Let f be the computable function whose existence is guaranteed by Theorem 7. Let $S=\{n \mid f(n)=1\}$ be the recursive set having f as characteristic function. Denote $g(\omega_S)=m_o$. The pair (L,ω_S) clearly computes the function f. By the definition of the length of computation function F_L and by (9) and (10) we have

$$F_L(n) < p(n,m_o)$$

for all n, contrary to (8). Thus L is undecidable.

Remark 5. Our proof actually shows that for every decidable system L there exists a recursive set S with a decision problem which is more difficult than all decision problems of all recursive sets representable in L.

B I B L I O G R A P H Y

1. S.C. Kleene, Introduction to Metamathematics, Van Norstrand Co., New York, 1952.

2. E. Post, Recusively enumerable sets of positive integers and their decision problems, Bull. Amer. Math. Soc., vol. 50 (1944), pp. 284-316.

3. P.C. Rosenbloom, The Elements of Mathematical Logic, Dover Publications, New York, 1950.

4. A. Tarski, A. Mostowski, R.M. Robinson, Undecidable Theories, North-Holland Co., Amsterdam, 1953.

5. M.O. Rabin, Speed of computation of functions and classification of recursive sets, Proceedings of the third Convention of Scientific Societies, Israel(1959), pp. 1-2.

L I S T O F R E C I P I E N T S

Assistant sec. of Def. for Res. and Eng.
Information Office Library Branch
Pentagon Building
Washington 25, D.C. (2 copies)

Armed Services Technical Information Agency
Arlington Hall Station
Arlington 12, Virginia (5 copies)

Chief of Naval Research
Department of the Navy
Washington 25, D.C.
Attn Code 437, Information Systems Branch
(2 copies)

Chief of Naval Research
Department of the Navy
Washington 25, D.C.
Attn Code 923

Director, Naval Research Laboratory
Technical Information Officer/Code 2000/
Washington 25, D.C. (6 copies)

Commanding Officer, Office of Naval Research
Navy #100, Fleet Post Office
New York, New York

Commanding Officer, O N R Branch Office
346 Broadway
New York 13, New York

Commanding Officer, O N R Branch Office
495 Summer Street
Boston 10, Massachusetts

Office of Technical Services
Technical Reports Section
Department of Commerce
Washington 25, D.C.

Chief, Bureau of Ships
Department of the Navy
Washington 25, D.C.
Attn Code 280

Chief, Bureau of Ships
Department of the Navy
Washington 25, D.C.
Attn Code 677

Chief, Bureau of Ships
Department of the Navy
Washington 25, D.C.
Attn Code 684

Chief, Bureau of Ships
Department of the Navy
Washington 25, D.C.
Attn Code 686

Chief, Bureau of Ships
Department of the Navy
Washington 25, D.C.
Attn Code 687E

Chief, Bureau of Ships
Department of the Navy
Washington 25, D.C.
Attn Code 687F

Chief, Bureau of Ships
Department of the Navy
Washington 25, D.C.
Attn Code 687G

Naval Ordance Laboratory
White Oaks
Silver Spring 19, Maryland
Attn Technical Library

David Taylor Model Basin
Washington 7, D.C.
Attn Technical Library

Chief, Bureau of Ordance
Department of the Navy
Washington 25, D.C.

aval Electronics Laboratory
an Diego 52, California
ttn Technical Library

Naval Ordance Laboratory
Corona, California
Attn Robert Conger, Electr. and Mag. Div

niversity of Illinois
ontrol Systems Laboratory
rbana, Illinois
ttn D. Alpert

niversity of Illinois
igital Computer Laboratory
rbana, Illinois
ttn Dr. R.E. Meagher

ir Force Office of Scientific Research
ashington 25. D.C.

ir Force Cambridge Research Center
aurence C. Hanscom Field
edford, Massachusetts
ttn Electronic Research Directorate
ibrary

echnical Information Officer
S Army Signal Research & Dev. Lab.
orth Monmouth, New Jersey
ttn Data Equipment Branch

ommanding Officer
iamond Ordance Fuze Laboratories
ashington 25, D.C.
ttn Tech. Ref. Section,/Ordtl 06.33/

ffice of Ordance Research
ox CM, Duke Station
urham, North Carolina

irector
ational Security Agency
ort Geo. G. Meade, Maryland
ttn Chief, Remp

Naval Proving Ground
Dahlgren, Virginia
Attn Naval Ordn. Computation Center

National Bureau of Standards
Washington 25, D.C.
Attn Dr. S.N. Alexander

Aberdeen Proving Ground, BRL
Aberdeen Proving Ground, Maryland
Attn Chief, Computation Lab

Syracuse University
Electrical Eng. Dpt.
Syracuse 10, New York
Attn Dr. Stanford Goldman

Princeton University
Electrical Engrg. Dept.
Princeton, New Jersey
Attn Professor F.S. Acton

Burroughs Corporation
Research Center
Paoli, Pennsylvania
Attn A.J. Meyerhoff

Hycon Eastern, Inc.
75 Cambridge Parkway
Cambridge 42, Massachusetts
Attn Mr. J.E. Deturk

Lockheed Missile Systems Division
Sunnyvale, California
Attn J.P. Nash

Univ. of Michigan
Ann Arbor, Michigan
Attn Dept. of Philosophy, Prof. A.W.

National Science Foundation
Washington 25, D.C.
Attn Res. Info. Cntr. and Advisory
Serv. of Info. Processing (2 copie