XXVI. NEUROPHYSIOLOGY*

W. S. McCulloch	R. C. Gesteland	Diane Major
F. S. Axelrod	M. C. Goodall	L. M. Mendell
H. A. Baldwin	B. H. Howland	W. F. Pickard
P. O. Bishop	W. L. Kilmer	W. H. Pitts
M. Blum	K. Kornacker	Helga Schiff
J. E. Brown	W. J. Lennon	A. Taub
S. Frenk	J. Y. Lettvin	P. D. Wall

RESEARCH OBJECTIVES

1. Basic Theory

The general problem of reliable computation in vertebrate central nervous systems has been solved in principle.[1] Also, neurophysiologists are making rapid progress on the functional organization of specialized regions in such systems. But no one has yet reported a way for thinking effectively about how the brain stem reticular system performs its task of committing an entire organism to either one mode of behavior or another.

Our problem is to construct a theory for the reticular system[2-4] which is compatible with known neuroanatomy and neurophysiology, and which will lead to testable hypotheses concerning its operation.

Our first approach was through the theory of ordinary one-dimensional iterative logic nets.[5-7] But all of the crucial questions in this theory turned out to be recursively unsolvable when considered generally, and combinatorially intractable when particularized with the required degrees of dependency among the variables.

Our second approach was through the theory of coupled nonlinear oscillators, but we soon found that there is not nearly enough of the right kind of mathematics to be of much help in our problem.

All we can report at the moment is that we are embarked on a kind of iterative net statistical decision theory which is comprehensive, versatile, and penetrating enough to stand a reasonable chance of success.

W. L. Kilmer, W. S. McCulloch

References

1. Cf. W. S. McCulloch, Biological Computers, IRE Trans., Vol. EC-4, No. 3, pp. 190-192, 1957; M. A. Arbib, Brains, Machines and Mathematics (McGraw-Hill Publishing Company, New York, in press); S. Winograd and J. D. Cowan, Reliable Computation in the Presence of Noise (The M.I.T. Press, Cambridge, Mass., 1963).

2. M. Scheibel and A. Scheibel, Biological models for command automata, Mitre Report 55-3, First Congress on the Information System Sciences, November 1962.

3. H. Jasper and others (eds.), The Reticular Formation of the Brain, Proc. Henry Ford Hospital International Symposium (Little, Brown and Company, Boston, 1958).

*This work was supported in part by the Bell Telephone Laboratories, Inc.; The Teagle Foundation, Inc.; the National Science Foundation (Grant G-16526); the National Institutes of Health (Grants MH-04737-03 and NB-04897-01); the U.S. Air Force (Aeronautical Systems Division) under Contract AF33 (616)-7783; and the National Aeronautics and Space Administration (Grant NsG-496).

4. E. R. Ramey and D. S. O'Doherty (eds.), Electrical Studies on the Unanesthetized Brain (Harper and Row, New York, 1960).

5. F. C. Hennie III, Iterative Arrays of Logical Circuits (The M.I.T. Press, Cambridge, Mass., 1961).

6. W. L. Kilmer, On dynamic switching in one-dimensional logic networks (Information and Control, in press).

7. W. L. Kilmer, Topics in the theory of one-dimensional iterative networks (to be published in Information and Control).

2. Project Plans

(a) The Nature of Biological Membrane. For some time we have felt that the performance of cell membrane could be understood in terms of the logical consequences of two or three extremely simple and reasonable assumptions. We are now hopeful about the outcome of this work. But it appears that there are several elementary, but lengthy, experiments to be done with tracers on semipermeable membranes, and these must be finished before we are willing to commit ourselves. The development is only distantly related to, and is also more transparent than, the current theories based on irreversible thermodynamics.

W. F. Pickard, J. Y. Lettvin

(b) The Nature of Form-Function Relations in Neurons. After Lettvin and Maturana proposed certain anatomical features as underlying the operation of retinal ganglion cells in the frog, the question arose as to whether it would be possible to test the notion. During this past year we discovered that the use of time constants of adaptation suggested the combinatorials that were proposed. This finding has done much to hearten us to make a more exact statement of connectivity, and we are now engaged in setting up a quantitative study.

S. Frenk, J. Y. Lettvin

(c) The Character of Certain Receptor Processes. It was suggested by Lettvin, in his analysis of color vision, that some receptors may have to be characterized by two variables at their outputs. For example, imagine a receptor like a rod that has a photosensitive pigment attached to the membrane. Let every patch of membrane with pigment on it have an equal effect at the output. Let a pigment molecule, on receipt of a quantum, first change in such a way as to open a channel for ions not at equilibrium with the membrane potential, thus causing a current flow. Thereafter the pigment further changes and either moves away from the membrane or becomes attached in a different way so as to open a channel only for ionic species at equilibrium with the membrane potential. In this way, the signal imposed by light at any instant is attenuated by a weighted integral of the amount of light shining in the immediate past (there is also a restoration process) and this process yields a total signal current which is logarithmically, or quasilogarithmically, related to the average intensity of light. Such a receptor process has two degrees of freedom. The idea that certain stimuli increase signal current while others attenuate that current by division is useful when one suspects combinatorials of stimuli to be taken at the very input. Some of the results of Gesteland on the olfactory mucosa suggest that such a process occurs there. We expect to examine this in some detail. A preliminary report has been prepared.

R. C. Gesteland, W. H. Pitts, J. Y. Lettvin

(d) Properties of Cerebellar Cells. The writer reported in Quarterly Progress Report No. 69 (pages 241-246) on the curious combinatorials of stimuli necessary to excite cells in the cerebellum of the frog. This research has continued and the results will be presented as a doctoral dissertation in 1964.

F. S. Axelrod

(e) Properties of Second-Order Vestibular Neurons in Frogs. The ganglion cells of the three major vestibular nuclei are related to the firing pattern of the different vestibular branches in definite ways that are still difficult to define. We intend to try to define these functions.

Helga Schiff, J. Y. Lettvin

(f) Properties of Second-Order Olfactory Neurons in Frogs. There seems to be an indication that the transients recorded on the surface of the olfactory lobe are not only related to the firing of mitral and other cells but also to glomerular activity. At any rate, the slower transients look suspiciously as if they arise from primary processes. This is only a hint thus far, but will be pursued, since the difficulty of interpreting primary receptor processes may be eased by getting some notion of how the information appears aggregated at the junctions in olfactory lobe.

H. A. Baldwin, J. Y. Lettvin

(g) Is the Rapid Water Movement in Some Plants Attributable to an Ion Pump? The current explanation of the very quick changes in turgor of some cells in Mimosa is that the osmotic pressure changes are brought about by the breaking up of starch. We suspect that an alternative is the movement of water by something like a potassium pump. This idea will be subjected to test during the early part of 1964 by Barbara Pickard.

J. Y. Lettvin

(h) Color Vision. The views reported in part in Quarterly Progress Report No. 70 (pages 327-337) will be carried farther by psychological studies, rather than by physiological experiments on monkeys.

J. Y. Lettvin

(i) Instrumentation Projects. Divers instruments will be built as the occasion arises, if the research demands them and industry is incapable of furnishing them. (Barbara Pickard will be connected with this project.)

H. A. Baldwin, J. Y. Lettvin

(j) Models of Electrochemical Processes. There is some reason to hope that some of the theory of weak interactions may be useful in a general theory of strong electrolytes. This idea will be pursued to test for adequacy.

W. H. Pitts

(k) Physical Optics. There are enough tricks of design left in physical optics that one can say the field is far from exhausted. This is particularly true when one wants to deal with the dioptrics of living eyes. Some of our results have appeared in Quarterly Progress Reports No. 67 (pages 197-204) and No. 71 (pages 267-273), and we intend to look for still more.

B. H. Howland

3. Problems of Sensory Projection Pathways

During the past year, we have concentrated on two major lines of approach to the problems of cutaneous sensory mechanisms. The first has dealt with the control system situated about the first central synapse where nerve fibers from the skin converge on cells in the dorsal part of the spinal cord. We have shown that the very small cells scattered throughout the region of these synapses and which make up the substantia gelatinosa are involved in modulating the transmission of impulses across this first junction. This censorship of arriving nerve impulses is affected by previous activity in the same pathway, by activity in neighboring areas of skin, by intense activity in distant areas, particularly in paws and face, and by stimulation of the cerebellum, mid-brain, pons, and medulla. The censorship mechanism seems to be in continuous action, and we believe that it is best studied by steady stimuli, rather than by sudden brief changes in the environment. The mechanisms that we have seen in the cat would predict interactions between various types of skin stimuli, and we have carried out concomitant experiments on man to examine these hypotheses. These psychological experiments have shown that there is a most interesting interaction in man between light-pressure stimuli and electrical stimulation. We have published some of this work in Brain and in the Journal of Physiology, and two other papers will appear in the latter journal in 1964. In the coming year, we shall pursue the study of the censorship mechanism in an attempt to find something of its role in the normal functioning of the animal.

Our second line of approach is an attempt to discover the language used by the skin in telling the brain about the location of the stimulus. We are studying two reflexes that require the motor mechanisms to know the exact location of the stimulus. The first is the scratch or swipe reflex, and the second is the eye blink. We are studying the pathways over which the information is carried both in normal animals and in frogs and salamanders who have been operated on in their youth. If dorsal and ventral skin are reversed in the tadpole, the scratch reflex of the adult frog is aimed at the embryological position of the skin, and not at its actual position, so that it is evident that some message is going from skin to central nervous system which tells the nature of the skin rather than its position. We hope to discover the nature of this message by microelectrode studies of the cord. Similar work is being done on amphibia in which an additional eye is implanted on the head. The extra eye will generate a blink reflex in the normal eye if it is touched, and so we know that nerves are somehow capable of telling the brain that they are in cornea and not in ordinary skin. This problem has been studied in normal frogs; a paper based on Karl Kornacker's doctoral dissertation has appeared in Experimental Neurology, and this work will continue.

A. Taub, K. Kornacker, Diane Major, P. D. Wall

A. ALGORITHMIC THEORIES OF GROWTH AND DIFFERENTIATION

This report is a summary of a paper that has been submitted for the Biophysical Society Annual Meeting to be held in Chicago, February 26-28, 1964.

The analogy of growth and differentiation to the unfolding of a computer program is obvious. No programs yet devised, however, satisfy the essential requirement of a biological model, which is functional stability in the face of a fluctuating environment. This poses a certain antithesis: on the one hand, analog systems possess only a local stability; on the other, digital systems can, by functional redundancy,[1] achieve stability in the large.

A model combining these features closely reflects what is known about cellular differentiation and morphogenesis. That is, each cell contains the basic genetic code

represented by the substitution rules R of Thue[2] associative system, together with current word S_i, and address A_i (for next application of R). S_i determines the rate constants of the cell metabolism regarded as a nonlinear system.[3] Finally, the concentrations of metabolites operate a threshold control system CS which determines A_i and the next application of R. This model is computable, but its main value at present would appear to be as a conceptual framework in which to discuss problems of epigenesis.

M. C. Goodall

References

1. W. S. McCulloch, The Stability of Biological Systems, Brookhaven Symp. 10, 207 (1957).

2. M. Davis, Computability and Unsolvability (McGraw-Hill Book Company, New York, 1958).

3. B. Goodwin, Temporal Organization in Cells (Academic Press, Inc., New York, 1963).

B. MEASURES ON THE COMPUTATION SPEED OF PARTIAL RECURSIVE FUNCTIONS

Given sufficient time, a man can perform all of the operations of a modern digital computer. The value of the computer, therefore, lies in the speed with which it operates, not in the particular operations that it performs. The intent of this report is to define precisely this concept of speed of computation, and to derive some theorems that characterize it.

1. The Measure Function Φ

It is natural to associate with a computer program P_i both the function ϕ_i that it computes and a measure function Φ_i that might indicate the speed of the program. For example, $\Phi_i(x)$ might be the number of symbols printed or erased during the computation of $\phi_i(x)$ (by means of P_i), or the number of times that the computer changes state, or the number of seconds that elapse from the moment that the computer starts until the time at which it stops, or even the amount of tape used in the computation. From another point of view, the computation of $\phi_i(x)$ could be interpreted as a proof of the appropriate statement $\phi_i(x) = y$. $\Phi_i(x)$ would then be the length of a proof terminating in a string of the form $\phi_i(x) = y$.

2. Applications of the Measure Function Φ

It is evident that the measure function is useful in comparing the speed of two programs for the same function. If programs P_i and P_j serve to compute the same function

$\phi_i = \phi_j$ and if $\Phi_i(x) < \Phi_j(x)$ for all x, then P_i is a "quicker" program than P_j.

This measure is also linked with the difficulty of computation. For example, the function 2^{2^x} naturally takes more time to compute than the function 2x, since the answer takes so much longer to print, but a function with values 0 and 1 which takes as long to compute as 2^{2^x} must have difficulties inherent in its computation.

Fermat, ca. 1650, conjectured that $2^{2^x}+1$ is prime for all non-negative integers x. This he verified for x = 0 through x = 4, and, without further evidence, he challenged his comtemporaries to disprove it. Euler accepted the challenge, and, with characteristic cunning, proved that $2^{2^5}+1 = 641 \times 6,700,417$ is not a prime. Since his time, a number of persons have investigated this conjecture for x larger than 5. In all cases tried, they have found that $2^{2^x}+1$ has proper factors. Fermat's conjecture can be reformulated in terms of the function

$$f(x) = \begin{cases} 1 & \text{if } 2^{2^x}+1 \text{ is prime} \\ 0 & \text{otherwise.} \end{cases}$$

The evidence suggests the hypothesis: $f(x) = 0$ for $x \geq 5$, and if true, a quick program for the computation of f muxt exist. Such a program simply sees to it that an output 1 is printed for inputs x less than 5 and that a 0 is printed for inputs x greater than 5. If, on the other hand, it can be shown that every program that computes f is slower than the quick program suggested above, the hypothesis must be false. Hence the measure provides important information about this function. Unfortunately, the results of this report do not suffice to provide this information, but we take a step in the right direction by defining the measure function and deriving some of its properties.

3. M-Computers

For clarity, it is essential to distinguish between computers and their mathematical models, which we call M-Computers. There are to be thought of as ideal devices that can be programmed to compute any partial recursive function. We pick one such device here as standard: The standard M-Computer is a device equipped with a container for cards, a tape scanner, and a tape that is infinite in both directions. The tape is divided into squares along its length and the scanner can look at one square at a time. The device is equipped to print one of the symbols B(blank), 0, 1, ..., 9 on the square that it is examining and shift the tape to right or left by one square. The container can hold an arbitrarily large but finite number of cards, called the program. On each card is printed a single 5-tuple $\langle q_i, S_i, S_j, D, q_j \rangle$. The symbols q_i, i = 0, 1, 2, ... are internal states, S_i is one of the symbols B, 0, 1, ... 9, and D is a direction R(right) or L(left). When the device is in state q_i and scans the symbol S_i, it prints the symbol S_j, shifts

the tape to right or left as dictated by D, and changes its internal state to q_j. If the device is in state q_k and scans the symbol S_k, and if no card in the container has printed on it a 5-tuple of the form $\langle q_k, S_k, \ldots \rangle$, then the device stops. Any program is allowed, subject to the condition that any two cards must differ at either the first internal state q_i or the first symbol S_i.

We can associate with each program a partial recursive function ϕ as follows: The program is placed in the container, an input integer x is written in radix 10 on the tape, the scanner is placed over the rightmost digit of x, and the device is put in state q. The device then operates in accordance with the instructions printed in the program. If it never stops, we say that the function $\phi(x)$, which it computes, diverges. If it does stop, we let $\phi(x)$ be the integer that remains on the tape after all B's are cancelled.

We can effectively list the programs P_0, P_1, ... for a standard M-Computer, their associated partial recursive functions ϕ_0, ϕ_1, ... (ϕ_i is computed by P_i), and their measure functions Φ_0, Φ_1, ... defined by the statement: $\Phi_i(x)$ diverges if $\phi_i(x)$ diverges, and $\Phi_i(x)$ is the number of seconds required to compute $\phi_i(x)$ if $\phi_i(x)$ converges.

The standard M-Computer with a program in its container is a Turing machine.

A basic result of recursive function theory states that for a large class of idealized computers, a function that is computable by one such device is computable by all others. Hence it is unnecessary for most authors to distinguish among them. It is essential for us to make this distinction, however, since we are concerned with measure functions, and since the particular measure function associated with a program must depend on our choice of computer. In fact, we shall define an M-Computer abstractly in terms of the measure functions associated with its programs.

DEFINITION: An M-Computer ${}^{\xi}C$ is a list of pairs of partial recursive functions $(\phi_0, {}^{\xi}\Phi_0)$, $(\phi_1, {}^{\xi}\Phi_1)$, $(\phi_2, {}^{\xi}\Phi_2)$ ... where ϕ_0, ϕ_1, ϕ_2, ... is the enumeration of all partial recursive functions as determined by the standard M-Computer,[1] and ${}^{\xi}\Phi_0$, ${}^{\xi}\Phi_1$, ${}^{\xi}\Phi_2$, ..., is an enumeration of partial recursive functions called measure functions, with the properties:

(i) For all i and x, $\phi_i(x)$ converges iff ${}^{\xi}\Phi_i(x)$ converges.

(ii) There exists a total recursive function α such that for all i and x

$$\alpha(i,x,y) = \begin{cases} 1 & \text{if } {}^{\xi}\Phi_i(x) = y \\ 0 & \text{otherwise.} \end{cases}$$

NOTATION: We use f, g, and h to denote arbitrary partial recursive functions, and the symbol ϕ_i to denote the i^{th} partial recursive function in our standard listing. To every partial recursive function f there corresponds infinitely many different i's for which $\phi_i = f$. Given such an i, we may sometimes write f_i for ϕ_i and ${}^{\xi}F_i$ for ${}^{\xi}\Phi_i$ in order to emphasize that ϕ_i equals f.

We now give some examples to indicate how the measure function may be interpreted.

EXAMPLE 1: We have defined $\Phi_i(x)$ to be the number of seconds required to compute $\phi_i(x)$ on the standard M-Computer. Property (i) of the measure functions states that $\phi_i(x)$ converges iff its computation takes a finite time ($\Phi_i(x)$). Similarly, property (ii) states that it is effectively possible to determine whether or not $\phi_i(x)$ converges in y seconds. One begins the computation of $\phi_i(x)$ with stop watch in hand; if the computation takes exactly y seconds, we let $a(i,x,y) = 1$; otherwise, we let $a(i,x,y) = 0$.

EXAMPLE 2: The measure function might measure the number of squares of tape used in the computation. Take the standard M-Computer and let

$$\xi_{\Phi_i(x)} = \begin{cases} y & \text{if } \phi_i(x) \text{ converges and y squares of tape are used in the computation} \\ \text{divergent} & \text{if } \phi_i(x) \text{ diverges.} \end{cases}$$

Clearly, $\phi_i(x)$ converges if and only if $\xi_{\Phi_i(x)}$ converges. A simple argument based on the number of possible machine tape configurations that involve a fixed finite amount of tape proves that a is total recursive. Hence ξC is an M-Computer.

EXAMPLE 3: Begin the computation of $\phi_i(x)$ with the standard M-Computer. Let $\xi_{\Phi_i(x)}$ be the number of symbols required to write the program P_i, plus the number of state changes during the computation of $\phi_i(x)$, plus the number of squares of tape used. Then ξC is an M-Computer.

EXAMPLE 4: Let C' be some characterization other than the standard M-Computer. Denote the programs and corresponding partial recursive functions of C' by

$$P'_0, P'_1, P'_2, \ldots$$

$$\phi'_0, \phi'_1, \phi'_2, \ldots .$$

Under what conditions may we assume that $\phi_0 = \phi'_0$, $\phi_1 = \phi'_1$, ... ? If the enumeration theorem and the s-m-n theorem[2] are to hold true for C', it can easily be shown that there must exist total recursive functions g and h such that $\phi'_{g(i)} = \phi_i$ and $\phi'_i = \phi_{h(i)}$ for all i. If, in addition, there exists a total recursive function k such that $\phi_{k(i)} = \phi_i$ and $k(i) > i$ for all i, then g and h can be taken to be 1-1 recursive functions. By means of a Schröder-Bernstein type of proof, it can then be shown that there exists a 1-1 onto total recursive function f such that $\phi'_{f(i)} = \phi_i$ for all i. Without loss of generality, we may then assume that f is the identity function. It follows, we hope, that the choice of a standard M-Computer in the definition given above is not too demanding.

In order to compute $\xi_{\Phi_i(x)}$, we must first find a partial recursive function $\phi_j = \xi_{\Phi_i}$, since we have a program for computing ϕ_j but none for computing ξ_{Φ_i}. Our first theorem

asserts that there is an effective procedure for going from an index i for a measure function to an index j for the equivalent partial recursive function in our standard listing.

THEOREM 1: To each M-Computer ${}^{\xi}C$ there corresponds a total recursive function β such that ${}^{\xi}\Phi_i = \phi_{\beta(i)}$ for all i.

PROOF: We begin by defining a function $f(i,x) = {}^{\xi}\Phi_i(x)$. Formally,

$$f(i,x) = \begin{cases} y & \text{if } a(i,x,y) = 1 \\ \text{divergent} & \text{otherwise.} \end{cases}$$

Since a is a total recursive function, it is effectively possible to compute f, hence by Church's thesis, f is a partial recursive function. The s-m-n theorem ensures the existence of a total recursive function β such that $\phi_{\beta(i)}(x) = f(i,x)$ for all i and x. Therefore $\phi_{\beta(i)} = {}^{\xi}\Phi_i$ for all i, and β is the desired total recursive function. Q.E.D.

Since the recursive functions are countable, it follows immediately that the class of M-Computers is also countable.

4. A Theorem by Rabin

Suppose that you want to find a function f such that no matter what program you choose to compute it, the calculation of f(x) always takes more that 2^x seconds. All that you need to do is to pick a function f with values so large that it takes 2^x seconds just to print the answer. A similar method can be used to find a function that takes more than g(x) seconds to compute, g being any total recursive function. A nontrivial problem, however, would be to find a function f with values 0 and 1 such that every program for f necessarily takes more than g(x) seconds to compute for almost all x (for all x greater than some integer x_o). (Question: Why for almost all x?) By means of an interesting diagonalization process, M. O. Rabin proved the existence of such 0-1 valued functions.[3] His theorem, with a somewhat different proof, is reproduced here. The process involved is an essential feature of most of the proofs in this report.

THEOREM 2 (Rabin): Let C be an M-Computer and let g be any total recursive function. Then there exists a total recursive function f with values 0 and 1 such that for every index i for f, ${}^{\xi}F_i(x)$ exceeds g(x) for almost all x.

PROOF: We construct f: First, a Gödel table is made which contains the symbol $\phi_r(c)$ in the intersection of row r and column c.

	0	1	2	3	4
0	$\phi_0(0)$	$\phi_0(1)$	$\phi_0(2)$	$\phi_0(3)$	~~$\phi_0(4)$~~
1	$\phi_1(0)$	$\phi_1(1)$	~~$\phi_1(2)$~~	~~$\phi_1(3)$~~	~~$\phi_1(4)$~~
2	$\phi_2(0)$	$\phi_2(1)$	$\phi_2(2)$	$\phi_2(3)$	$\phi_2(4)$
3	$\phi_3(0)$	$\phi_3(1)$	$\phi_3(2)$	$\phi_3(3)$	$\phi_3(4)$
4	$\phi_4(0)$	$\phi_4(1)$	$\phi_4(2)$	$\phi_4(3)$	$\phi_4(4)$

Then all entries $\phi_i(x)$ that converge in $g(x)$ seconds at most are circled, that is, $\phi_i(x)$ is circled if ${}^{\xi}\Phi_i(x) \leq g(x)$. This circling can be done effectively, since ${}^{\xi}\Phi_i(x) \leq g(x) \leftrightarrow \alpha(i,x,y) = 1$ for some y less than or equal to $g(x)$, and g is total recursive.

The next stop is to go to column 0: If $\phi_o(0)$ is circled, check it, cancel the remaining part of row 0, and go to column 1. If $\phi_o(0)$ is not circled, go directly to column 1. In general, when you reach any column, x, check the first circled uncancelled entry in that column, $\phi_i(x)$, such that $i \leq x$. Then cancel all entries to the right of $\phi_i(x)$. If no such entry exists, do nothing to column x. Go to column $x+1$. This check procedure is clearly effective. Moreover, a row with infinitely many circled entries must have 1 and only 1 checked entry.

To compute $f(x)$, first determine if one of the entries $\phi_o(x)$, $\phi_1(x)$, ..., $\phi_x(x)$ is checked (by the procedure above, one of these entries at most can be checked). If, say, $\phi_i(x)$ is checked, it must converge (since $\phi_i(x)$ converges iff ${}^{\xi}\Phi_i(x)$ converges, and ${}^{\xi}\Phi_i(x) \leq g(x)$). Compute $\phi_i(x)$ and let

$$f(x) = \begin{cases} 1 & \text{if } \phi_i(x) = 0 \\ 0 & \text{otherwise.} \end{cases}$$

If none of the entries is checked, let $f(x) = 1$. Clearly, f is a total recursive function. To see that it is the desired function, suppose that i is an index for f and that, to the contrary, there exist infinitely many x such that ${}^{\xi}F_i(x) \leq g(x)$. Then, according to the procedure given above, row i must have a single checked entry $\phi_i(y)$. But, by definition of f, $f(y) \neq \phi_i(y)$, hence i cannot be an index for f, which is a contradiction. Therefore f must be the desired function. Q.E.D.

The next theorem and its proof is an immediate generalization of Rabin's theorem to all partial recursive functions. The statement of this theorem will be needed as a lemma subsequently in this report.

THEOREM 3: Let ${}^{\xi}C$ be an M-Computer and let g be a partial recursive function. Then there corresponds to g a 0-1 valued partial recursive function f such that $f(x)$ converges iff $g(x)$ converges; if j is any index for f, then for almost all x ${}^{\xi}F_j(x) > g(x)$ whenever $g(x)$ converges. Moreover, there is a total recursive function, γ, which takes any index i for g into an index $\gamma(i)$ for f.

PROOF: Fix an integer i and construct a list of pairs of integers according to the following procedure (these pairs correspond to the checked entries in the proof of Rabin's theorem)[4]:

Stage 0: If ${}^{\xi}G_i(0) = 0$ (i.e., $\alpha(i,0,0) = 1$) and ${}^{\xi}\Phi_o(0) \leq g(0)$ (${}^{\xi}G_i(0) = 0 \rightarrow g_i(0)$ converges), put $(0,0)$ in the list; otherwise, do nothing. Go to stage 1.

Stage n = (p,q): Determine whether or not ${}^{\xi}G_i(p) = q$. If not, go to stage $n+1$. If so, then $g_i(p)$ converges. See if any of the function values ${}^{\xi}\Phi_o(p)$, ${}^{\xi}\Phi_1(p)$, ... ${}^{\xi}\Phi_p(p)$

are less than or equal to $g_i(p)$. (This can be done with the α-function.) If not, go to stage n+1. If so, then ${}^{\xi}\Phi_{s_1}(p), \ldots, {}^{\xi}\Phi_{s_t}(p)$ $(s_1 < \ldots < s_t \leq p)$ will be less than or equal to $g_i(p)$. If all of the pairs $(s_1, z_1), \ldots, (s_t, z_t)$ appear in the list thus far constructed for some values of $z_1, \ldots, z_t$, go to stage n+1. If not, pick the smallest integer s_k such that (s_k, z_k) does not appear in the list thus far constructed for any value of z_k. (This integer s_k can be chosen effectively, since the list thus far constructed is finite.) Put (s_k, p) in the list. Go to stage n+1.

The list of pairs obtained in this way is obviously recursively enumerable. Note that for all s, s', p, p', if (s, p) appears in the list, then (s, p') and (s', p) do not appear in the list: (s, p') cannot appear for obvious reasons; (s', p) cannot appear since to do so, it must appear in stage $n = \langle p, q \rangle$ where ${}^{\xi}G_i(p) = q$, but this is precisely the stage in which (s, p) must be placed in the list, and at each stage at most one pair of integers can be placed in the list.

We now give an effective procedure for computing a partial recursive function h of two variables i and x from which we shall obtain the function f by an application of the s-m-n theorem. To compute h(i, p): First, compute $g_i(p)$. If it diverges, let h(i, p) diverge. If $g_i(p)$ converges, then ${}^{\xi}G_i(p) = q$ for some integer q. In this case, generate the list and see if (s, p) appears in it for some integer s. (If it does appear, then it must do so in stage $n = \langle p, q \rangle$; hence it is effectively possible to determine if (s, p) is in the list for some s.) If (s, p) does <u>not</u> appear in the list for some s, let h(i, p) = 1. If it does appear, then $\phi_s(p)$ converges and ${}^{\xi}\Phi_s(p) < g_i(p)$ (by construction of the list) and s is unique (for if $t \neq s$, then (t, p) does not appear in the list). Let

$$h(i,p) = \begin{cases} 0 & \text{if } \phi_s(p) \neq 0 \\ 1 & \text{if } \phi_s(p) = 0. \end{cases}$$

Clearly, h is a partial recursive function, so that, by the s-m-n theorem, there exists a total recursive function γ such that $f_{\gamma(i)}(x) = h(i, x)$ for all i and x.

We assert that γ is the desired total recursive function of the theorem: If i is an index for a partial recursive function g, then $\gamma(i)$ is an index for a partial recursive function f such that, for all x, f(x) converges iff g(x) converges and f(x) = 0 or 1 if it converges, by definition of h. We want to prove that if j is any index for f, then ${}^{\xi}F_j(x) > g(x)$ for almost all x whenever g(x) converges. Suppose on the contrary that for infinitely many x, g(x) converges and ${}^{\xi}F_j(x) \leq g(x)$. Let $x_1, x_2, \ldots$ be the subset of these x that satisfy $x_k > j$. Then (j, x_k) does not appear in the list for any integer x_k, for otherwise $h(i, x_k) \neq f_j(x_k)$ (by definition of h and the fact that $g_i(x_k)$ converges), which is a contradiction; thus, as we said, (j, x_k) does not appear in the list. But at stage $n = \langle x_k, {}^{\xi}G_i(x_k) \rangle$, (j, x_k) is a candidate for the list, since ${}^{\xi}F_j(x_k) \leq g(x_k)$. Hence there exists an integer s less than j such that ${}^{\xi}\Phi_s(x_k) \leq g(x_k)$ and (s, z) does not appear in the list

up to that point for any integer z; this must be true for $x = x_1, x_2, \ldots$. But there can be at most j such integers s, namely $s = 0, \ldots, s = j - 1$. But there are an infinite number of integers x_k and thus eventually the integers s must be exhausted, which is a contradiction. Q.E.D.

5. Bounds on the Measure Function

Suppose that f(x) is defined in terms of p(x), q(x), ... Then one might expect that the time that it takes to compute f with a "reasonable" program (one that does not do complicated and unnecessary intermediate calculations) is bounded by a function of the time that it takes to compute p(x), q(x), ... For example, the enumeration theorem[2] asserts the existence of a partial recursive function f defined by $f(<i, x>) = \phi_i(x)$.[4] Can we bound the time that it takes for a reasonable program to compute f by some function of the time that it takes to compute each of the ϕ_i? We show that this is possible on the standard M-Computer with a particular program that computes f(n) in the following way.

1. Decode n to the form $n = <i, x>$.

2. Enumerate the list of machine programs until the i^{th} machine program P_i is reached and erase everything on the tape except the integer x and the instructions P_i.

3. Begin the computation of $\phi_i(x)$ by operating on x more or less as P_i would. Naturally, this requires that occasional reference be made to the instructions P_i that are printed on the tape.

4. When the solution $\phi_i(x)$ is reached, erase the symbols representing the machine program P_i and leave only the symbols representing this output.

Suppose that P_j is this program for f. Then the contributions to $F_j(n)$ made by the four steps outlined above are:

1. $g'(n)$ = time required to decode n as $<i, x>$.

2. $g''(i)$ = time required to enumerate the programs up to P_i.

3. $g'''(\Phi_i(x))$ = time required to compute $\phi_i(x)$ with occasional glances at P_i. (g''' is total, although $g'''(\Phi_i(x))$ will diverge if $\phi_i(x)$ diverges.)

4. $g''''(i)$ = time required to erase P_i.

Hence $F_j(<i, x>) = g'(<i, x>) + g''(i) + g'''(\Phi_i(x)) + g''''(i)$, where $g', \ldots, g''''$ are total recursive functions. More accurately,

$$F_j(i, x) = \begin{cases} g'(<i, x>) + g''(i) + g'''(\Phi_i(x)) + g''''(i) & \text{if } \phi_i(x) \text{ converges} \\ \text{divergent} & \text{otherwise.} \end{cases}$$

Our next theorem deals with an arbitrary M-Computer ${}^{\xi}C$; it proves that if j is any index for the function f, then there exists a total recursive function g determined by the index j such that ${}^{\xi}F_j(<i, x>) \leq g(i) + g(x) + g({}^{\xi}\Phi_i(x))$ for all i and x. An unexpected feature of this theorem is that it holds even when the program for f is "unreasonable."

LEMMA: Let h be a total recursive function of n variables. Then there exists a monotonically increasing total recursive function g of one variable such that $h(x_1, \ldots, x_n) \leq g(x_1) + \ldots + g(x_n)$ for all non-negative integers $x_1, \ldots, x_n$.

PROOF: Trivial. Q.E.D.

THEOREM 4: Let ${}^{\xi}C$ be an M-Computer. Let j be an index for the function f defined by $f(<i,x>) = \phi_i(x)$ for all i and x. Then there exists a total recursive function g such that, for all i and x, ${}^{\xi}F_j(<i,x>) \leq g(i) + g(x) + g\left({}^{\xi}\Phi_i(x)\right)$ whenever $\phi_i(x)$ converges.

PROOF: We shall prove the existence of a total recursive function h such that for all i and x, if $\phi_i(x)$ converges, then ${}^{\xi}F_j(<i,x>) = h\left(i, x, {}^{\xi}\Phi_i(x)\right)$. The existence of the desired g then follows from the lemma. To compute h(i, x, z), first, determine, by means of the α-function, whether or not ${}^{\xi}\phi_i(x) = z$. If not, let h(i, x, z) = 0. Otherwise, compute ${}^{\xi}F_j(<i,x>)$ (it must converge, since ${}^{\xi}\Phi_i(x)$ convergent $\rightarrow \phi_i(x)$ convergent $\rightarrow$ $f_j(<i,x>)$ convergent $\rightarrow {}^{\xi}F_j(<i,x>)$ convergent), and let $h(i,x,z) = {}^{\xi}F_j(<i,x>)$. This h is clearly the desired total recursive function. Q.E.D.

We shall now give three other theorems that are very much like the one above. Our purpose is to provide a method, although the statements are interesting in themselves. All theorems are proved for an arbitrary M-Computer ${}^{\xi}C$.

THEOREM 5: Let x_o be a non-negative integer. Let s be a total recursive function such that for all i, j and x

$$f_{s(i,j)}(x) = \begin{cases} \phi_i(x) & \text{if } x \leq x_o \\ \phi_j(x) & \text{if } x > x_o. \end{cases}$$

Then there exists a total recursive function g such that for almost all x greater than x_o, ${}^{\xi}F_{s(i,j)}(x) \leq g(x) + g(j) + g\left({}^{\xi}\Phi_j(x)\right)$ whenever $\phi_j(x)$ converges.

PROOF: Note that the existence of a total recursive function s follows from an application of the s-m-n theorem to the function

$$r(i,j,x) = \begin{cases} \phi_i(x) & \text{if } x \leq x_o \\ \phi_j(x) & \text{if } x > x_o. \end{cases}$$

We first prove the existence of a total recursive function h such that for almost all x ${}^{\xi}F_{s(i,j)}(x) \leq h\left(x, j, {}^{\xi}\Phi_j(x)\right)$ whenever $\phi_j(x)$ converges. To compute h(x, j, z), proceed as follows: If $x \leq x_o$, let h(x, j, z) = 0. If $x > x_o$, determine whether or not ${}^{\xi}\Phi_j(x) = z$. If not, let h(x, j, z) = 0. Otherwise, it follows that $\phi_j(x)$ converges and, by definition of s, $f_{s(i,j)}(x)$ converges for all i. Let

$$h(x,j,z) = \text{maximum}\left[{}^{\xi}F_{s(o,j)}(x), {}^{\xi}F_{s(i,j)}(x), \ldots {}^{\xi}F_{s(x,j)}(x)\right].$$

Clearly, h is a total recursive function, and, if $x \geq \max[i, x_0]$, then $h\left(x, j, {}^{\xi}\Phi_j(x)\right) \geq {}^{\xi}F_{s(i,j)}(x)$ whenever $\phi_j(x)$ converges. The existence of the desired function g follows from the existence of h by the lemma. Q.E.D.

THEOREM 6: Let s be a total recursive function such that $\phi_{s(i)}(x) = \phi_{\phi_i(x)}(x)$ for all i and x. Then there exists a monotonically increasing total recursive function g such that for all i and x

$$ {}^{\xi}\Phi_{s(i)}(x) \leq g(x) + g(i) + g\left({}^{\xi}\Phi_i(x)\right) + g\left({}^{\xi}\Phi_{\phi_i(x)}(x)\right) $$

wherever $\phi_{s(i)}(x)$ converges.

PROOF: The existence of the total recursive function s follows from a single application of the s-m-n theorem. First, we prove the existence of a total recursive function h such that for all i and x ${}^{\xi}\Phi_{s(i)}(x) = h\left(x, i, {}^{\xi}\Phi_i(x), {}^{\xi}\Phi_{\phi_i(x)}(x)\right)$ whenever $\phi_{s(i)}(x)$ converges. To compute $h(x, i, y, z)$, determine whether or not ${}^{\xi}\Phi_i(x) = y$. If not, let $h(x, i, y, z) = 0$. Otherwise, determine whether or not ${}^{\xi}\Phi_y(x) = z$. If not, let $h(x, i, y, z) = 0$. Otherwise, compute ${}^{\xi}\Phi_{s(i)}(x)$, which must converge because ${}^{\xi}\Phi_{\phi_i(x)}(x)$ converges to z. Let $h(x, i, y, z) = {}^{\xi}\Phi_{s(i)}(x)$. Obviously, h is a total recursive function and the existence of the desired total recursive function g follows. Q.E.D.

THEOREM 7: Let γ be the function defined in Theorem 3. Then there exists a monotonically increasing total recursive function g such that for all i and x

$$ {}^{\xi}\Phi_{\gamma(i)}(x) \leq g(x) + g(i) + g\left({}^{\xi}\Phi_i(x)\right) $$

whenever $\phi_i(x)$ converges.

PROOF: Let

$$ h(x, i, z) = \begin{cases} 0 & \text{if } {}^{\xi}\Phi_i(x) \neq z \\ {}^{\xi}\Phi_{\gamma(i)}(x) & \text{if } {}^{\xi}\Phi_i(x) = z. \end{cases} $$

h is total recursive and ${}^{\xi}\Phi_{\gamma(i)}(x) = h\left(x, i, {}^{\xi}\Phi_i(x)\right)$ whenever $\phi_i(x)$ converges. The existence of g follows immediately. Q.E.D.

6. Existence Theorems

The complexity of a function f often increases with x, so that a reasonable program for f computes f(x) more slowly than f(x-1) for all x. For example, the function $f(x) = 2^{2^x}$ seems to be of this type. If f is 0-1 valued, however, the time that it takes a program to compute it can often be drastically cut by a second program for infinitely many choices of x, and for these x the second program often computes f(x) more quickly than

f(x-1). For example, it seems that a reasonable program for the function

$$f(x) = \begin{cases} 1 & \text{if } 2^x+1 \text{ is prime} \\ 0 & \text{otherwise} \end{cases}$$

must require more time to compute f(x) than f(x-1), since the time required to compute 2^x+1 exceeds the time required to compute $2^{x-1}+1$. A theorem in number theory, however, states that 2^x+1 can be an odd prime only if $x = 2^y$ for some y; hence f(x) = 0 if $x \neq 2^y$. Clearly, this theorem speeds the computation of f(x) for $x \neq 2^y$, so that, for infinitely many x, f(x) can be computed more quickly than f(x-1). Nevertheless, it does not seem that every 0-1 valued total recursive function f can be simplified in this way. Intuition tells us that there exist 0-1 valued functions f with the property that every reasonable program for f must compute f(x) more slowly than f(x-1). We now confirm our suspicions.

THEOREM 8: To each total recursive function h there corresponds a total recursive function r with r(x) > h(x) for all x, and a 0-1 valued total recursive function f so that

i. If i is any index for f, then ${}^{\xi}F_i(x) > r(x)$ for almost all x.

ii. There exists an index k for f so that $r(x+1) > {}^{\xi}F_k(x) > r(x)$ for almost all x.

PROOF: Let g be the total recursive function of Theorem 7. First, we prove the existence of a total recursive function ϕ_j such that $\phi_j(x) > \max\left[h(x), g(x-1)+g(j)+g\left({}^{\xi}\Phi_j(x-1)\right)\right]$ for all x. Then we define p as follows:

$$p(i,x) = \begin{cases} h(0) + 1 & \text{if } x = 0 \\ 1 + \max\left[h(x), g(x-1)+g(i)+g\left({}^{\xi}\Phi_i(x-1)\right)\right] & \text{if } x > 0 \text{ and } \phi_i(x-1) \text{ converges} \\ \text{divergent} & \text{otherwise.} \end{cases}$$

Here, p is partial recursive, since h and g are total recursive, and the convergence of $\phi_i(x-1)$ implies the convergence of ${}^{\xi}\Phi_i(x-1)$. By the s-m-n theorem, there exists a total recursive function s such that $\phi_{s(i)}(x) = p(i,x)$ for all i and x. The recursion theorem[5,6] asserts the existence of an integer j such that $\phi_j = \phi_{s(j)}$; hence, $\phi_j(x) = p(j,x)$ for all x. To see that ϕ_j is the desired total recursive function, first note that $\phi_j(0) = h(0) + 1$. Assume that $\phi_j(y)$ converges for all $y < x$. Then by definition of p and the fact that ${}^{\xi}\Phi_j(x-1)$ converges, $\phi_j(x) = 1 + \max\left[h(x), g(x-1)+g(j)+g\left({}^{\xi}\Phi_j(x-1)\right)\right]$ must converge. Hence the existence of ϕ_j is proved.

Let $r = \phi_j$. Then r is total recursive and r(x) > h(x) for all x (by definition of ϕ_j) as desired. Let $\gamma(j)$ be the index of a function f. Then by Theorem 3 and the fact that r is total, f(x) = 0 or 1 for all x, and if i is any index for f, then ${}^{\xi}F_i(x) > r(x)$ for almost all x. This proves property (i). $\phi_j(x) > g(x-1) + g(j) + g\left({}^{\xi}\Phi_j(x-1)\right)$ (by definition of p) $\geq {}^{\xi}\Phi_{\gamma(j)}(x-1)$ (by Theorem 7) $= {}^{\xi}F_{\gamma(j)}(x-1)$ (by definition of f). Let $k = \gamma(j)$. Then

since $\phi_j = r$, $r(x) > {}^{\xi}F_k(x-1) > r(x-1)$ for all positive integers x. This proves property (ii) and completes the proof of this theorem. Q.E.D.

Note: The recursion theorem can be tricky and must be applied with care. In particular, the statement $\phi_j = \phi_{s(j)}$ does not imply that ${}^{\xi}\Phi_j = {}^{\xi}\Phi_{s(j)}$, since s(j) may be different from j.

A function rarely enjoys a unique quickest program for its computation. If the function is reasonably complex, no matter what program is chosen to compute it, another can be found which cuts the computation time in half for infinitely many x. Thus a program for f which takes x seconds to compute for infinitely many x could be replaced by another program that takes only x/2 seconds, and then again by another program that takes only x/4 seconds, and so on. As a simple example, suppose that we wish to know if an integer written in radix 10 is a palindrome. To solve this problem we write a program for the standard M-Computer which computes the function

$$f(x) = \begin{cases} 1 & \text{if } x \text{ is a palindrome} \\ 0 & \text{otherwise.} \end{cases}$$

The program is such that when the input integer is x = 3726854586273, the computer scans the rightmost digit 3 and goes down the tape to compare it with the leftmost digit 3, then back up to rightmost digit 7, and down to leftmost digit 7, and so on. After opposing digits have been compared, the computer prints an output 1. With a quicker program, the computer scans the rightmost digits 7 and 3 simultaneously, then goes down the tape to leftmost digits 3 and 7, then back up to digits 6 and 2, and so on, comparing the digits two at a time rather than one at a time. This program takes approximately one-half the time of the slower one for all palindromes. It can be shown, in fact, that no matter which program is chosen to compute this f, another can be given which computes f in one-half the time for almost all palindromes. One can further prove the existence of a total recursive function g with the property that to every program P_i for g there corresponds a quicker program P_j for which $G_i(x) > 2G_j(x)$ for almost all x. Can a stronger theorem be proved? In particular, can we find a function with the property that to every program P_i for f there corresponds a program P_j so much quicker that $F_i(x) > 2^{F_j(x)}$ for almost all x?

THEOREM 9: Let r be a total recursive function of two variables. Then there exists a total recursive function f with values 0 and 1 such that to every index i for f, there corresponds another index j for f such that ${}^{\xi}F_i(x) > r\left(x, {}^{\xi}F_j(x)\right)$ for almost all x.

PROOF: (1) The d and s functions.

Construct a Gödel table that contains the symbol $\phi_r(c)$ at the intersection of row r and column c. To compute d(i, x): First, compute $\phi_i(0)$, $\phi_i(1)$, ..., $\phi_i(x)$. If any of these diverge, let d(i, x) diverge. Otherwise, use the α-function to circle all entries

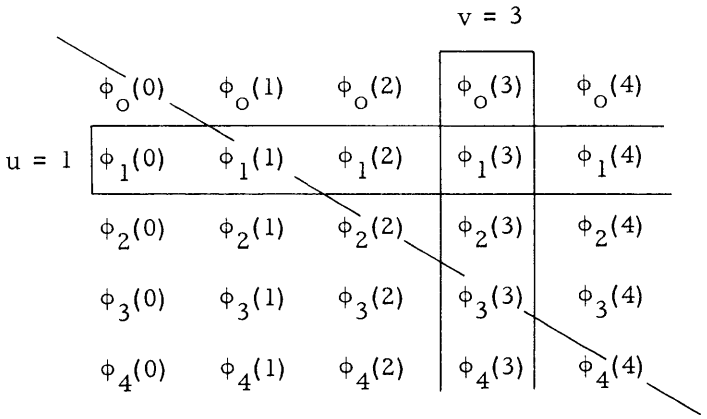
$\phi_j(x)$ in column x with $j \leq x$ and ${}^{\xi}\Phi_j(x) \leq \phi_i(x-j)$. Then check the first circled entry $\phi_k(x)$, if any, with the property that $\phi_k(y)$ has not been checked for $y < x$ (it is possible to determine if $\phi_k(y)$ has been checked for $y < x$, since $\phi_i(0), \ldots, \phi_i(x)$ converge). If none of the entries $\phi_o(x), \ldots, \phi_x(x)$ in column x are checked, let $d(i,x) = 0$. Otherwise (a single entry $\phi_k(x)$ is checked), let $d(i,x) = 0$ if $\phi_k(x) \neq 0$ and let $d(i,x) = 1$ if $\phi_k(x) = 0$. Clearly, d is partial recursive and if ϕ_i is total then d is total. Let s be a total recursive function which satisfies the equation $d(i,x) = \phi_{s(i)}(x)$ for all i and x (the existence of s is ensured by the s-m-n theorem). Note that if ϕ_i is total and if j is any index such that $\phi_j = \phi_{s(i)}$, then ${}^{\xi}\Phi_j(x) > \phi_i(x-j)$ for almost all x (for otherwise ${}^{\xi}\Phi_j(x) \leq \phi_i(x-j)$ for infinitely many $x = x_1, x_2, \ldots$; then in the computation of d, an entry $\phi_j(x_k)$ is checked and thus, $d(i,x_k) \neq \phi_j(x_k)$, which implies that $\phi_j \neq \phi_{s(i)}$ which is a contradiction).

(2) The e and t functions.

Define a function e of four variables x, u, v, i as follows:

CASE I $v \leq u$: Let $e(x,u,v,i) = e(x,u,u+1,i)$.

CASE II $v > u$: Construct a Gödel table and box row u and column v.



If $x < v$, let $e(x,u,v,i) = d(i,x)$. If $x \geq v$, compute $\phi_i(0), \phi_i(1), \ldots, \phi_i(x-u)$ and, if any of these diverge, let e(x,u,v,i) diverge. Otherwise, circle all entries $\phi_j(x)$ in column x with $j \geq u$ and $j \leq x$ and ${}^{\xi}\Phi_j(x) \leq \phi_i(x-j)$. Then check the first circled entry $\phi_k(x)$, if any, with the property that $\phi_k(y)$ has not been checked for $v \leq y < x$ during the computation of e(y,u,v,i) or for $y < v$ during the computation of d(i,y). If none of the entries $\phi_u(x), \ldots, \phi_x(x)$ is checked, let $e(x,u,v,i) = 0$. Otherwise (a single entry $\phi_k(x)$ is checked), let $e(x,u,v,i) = 0$ if $\phi_k(x) \neq 0$, and let $e(x,u,v,i) = 1$ if $\phi_k(x) = 0$. Clearly, e is partial recursive, and, if ϕ_i is total, then e is total. Let t be a total recursive function such that $e(x,u,v,i) = \phi_{t(u,v,i)}(x)$ for all x, u, v, and i. Note that

$$\phi_{t(o,o,i)}(x) = e(x,o,o,i) = d(i,x) = d_{s(i)}(x) \qquad \text{for all i and x.}$$

(3) Choose a total recursive 1-1 map of the integers onto the set of all 1-tuples, 2-tuples, 3-tuples, ... of integers. Let $\langle\langle a_o, \ldots, a_n \rangle\rangle$ denote the integer that

maps into $(a_o, \ldots, a_n)$. We now prove the existence of a total recursive function g such that for all x, u, v, and i

$$\xi\Phi_{t(u,v,i)}(x) = \begin{cases} g\left(x,u,v,i,\langle\!\langle {}^{\xi}\Phi_i(0), \ldots, {}^{\xi}\Phi_i(x-u)\rangle\!\rangle\right) & \text{if } \phi_i(0), \ldots, \phi_i(x-u) \\ & \text{converge} \\ \text{diverges} & \text{otherwise.} \end{cases}$$

To compute g(x, u, v, i, z), first, expand z according to the above-given coding as $z = \langle\!\langle a_o, \ldots, a_{x-u}\rangle\!\rangle$. Then, determine, by means of the α-function, whether or not ${}^{\xi}\Phi_i(0) = a_o, \ldots, {}^{\xi}\Phi_i(x-u) = a_{x-u}$. If not, let g(x, u, v, i, z) = 0. Otherwise, $\phi_{t(u,v,i)}(x)$ converges; thus let $g(x,u,v,i,z) = {}^{\xi}\Phi_{t(u,v,i)}(x)$.

(4) Let r be the function in the statement of the theorem and let g be the function of part (3). We now prove the existence of a total recursive function ϕ_{i_o} such that for all u and v and for almost all x

$$r\left[x, g\left(x,u,v,i_o, \langle\!\langle {}^{\xi}\Phi_{i_o}(0), \ldots, {}^{\xi}\Phi_{i_o}(x-u)\rangle\!\rangle\right)\right] \leq \phi_{i_o}(x-u+1).$$

Define h as follows:

$$h(i,o) = 0$$

$$h(i,z+1) = \begin{cases} \max\limits_{\substack{0\leq v\leq z \\ 0\leq w\leq z}} r\left[z+w, g\left(z+w, w, v, i, \langle\!\langle {}^{\xi}\Phi_i(0), \ldots {}^{\xi}\Phi_i(z)\rangle\!\rangle\right)\right] & \text{if } \phi_i(0), \ldots, \phi_i(z) \text{ converge} \\ \text{divergent} & \text{otherwise} \end{cases}$$

By the s-m-n theorem, there exists a total recursive function ρ such that $h(i,z) = \phi_{\rho(i)}(z)$ for all i and z. By the recursion theorem,[5] there exists an index i_o such that $\phi_{\rho(i_o)}(z) = \phi_{i_o}(z)$ for all z. Then $\phi_{i_o}(0) = 0$. Assume that $\phi_{i_o}(x)$ converges for $x \leq z$. Then $\phi_{i_o}(z+1)$ converges, since g and r are total recursive, and convergence of ϕ_{i_o} for $x \leq z$ implies convergence of ${}^{\xi}\Phi_{i_o}(x)$ for $x \leq z$. Therefore ϕ_{i_o} is total recursive. Fix the values of u and v; then, for $x > \max[2u, u+v]$, $\phi_{i_o}(x-u+1) \geq r\left[x, g\left(x,u,v,i_o, \langle\!\langle {}^{\xi}\Phi_{i_o}(0), \ldots, {}^{\xi}\Phi_{i_o}(x-u)\rangle\!\rangle\right)\right]$.

(5) We now show that for all u there exists a v such that $\phi_{t(u,v,i_o)} = \phi_{t(o,o,i_o)}$. Since ϕ_{i_o} is total recursive, so is $\phi_{t(o,o,i_o)}$. Recall that the computation of $\phi_{t(o,o,i_o)}(x)$ proceeds in two stages.

Stage 1: With respect to the Gödel table, certain entries in column x are circled and then one of these entries may be checked.

Stage 2: $\phi_{t(o,o,i_o)}(x)$ is made equal to zero if no entry in column x is checked;

otherwise, its value is made to depend on that of the checked entry.

If, in the Gödel table, row u is boxed, the number of entries above this row which are checked in the computation of $\phi_{t(o,o,i_o)}$ must be finite, since a row may contain at most one checked entry. Box the column v that lies to the right of the last checked entry above row u. Then by definition of t,

$$\phi_{t(o,o,i_o)}(x) = \phi_{t(u,v,i_o)}(x) \qquad \text{for all } x.$$

(6) We now define the function f in the statement of the theorem as $f(x) = \phi_{t(o,o,i_o)}(x)$ for all x. The function f is total recursive, since ϕ_{i_o}, and therefore $\phi_{t(o,o,i_o)}$, is total recursive. To prove the theorem, let i be any index for f. Let v be chosen according to part (5), so that $\phi_{t(i+1,v,i_o)} = \phi_{t(o,o,i_o)} = f$. Then

$$r\left[x, {}^{\xi}\Phi_{t(i+1,v,i_o)}(x)\right] = r\left[x, g\left(x, i+1, v, i_o, \langle\langle {}^{\xi}\Phi_{i_o}(0), \ldots, {}^{\xi}\Phi_{i_o}(x-i-1)\rangle\rangle\right)\right]$$

(according to part (3), since ϕ_{i_o} is total)

$$\leq \phi_{i_o}(x-i) \qquad \text{(according to part (4))}$$

$$< {}^{\xi}F_i(x) \qquad \text{(since } f(x) = \phi_{t(o,o,i_o)}(x) = \phi_{s(i_o)}(x) \text{ as pointed out in the last sentence of (2), and for reasons stated in the last sentence of (1)).}$$

for almost all x. Let $j = t(i+1, v, i_o)$. Then for almost all x, $r\left[x, {}^{\xi}F_j(x)\right] < {}^{\xi}F_i(x)$ as desired. Q.E.D.

7. Implications of Theorem 9

The implications of Theorem 9 are stated for the standard M-Computer, although they are true for any other M-Computer.

1. Let $r(x,y) = 2^y$. Then Theorem 9 asserts the existence of a total recursive function f such that to every index i for f there corresponds another index j such that $F_i(x) > 2^{F_j(x)}$ for almost all x. Hence it also asserts that to index j there corresponds an index k such that $F_i(x) > 2^{2^{F_k(x)}}$ and to k an index l such that $F_i(x) > 2^{2^{2^{F_i(x)}}}$, and so on. Hence there exists an infinite sequence of programs for f, starting with any program that one chooses so that each program in the sequence is followed by a much quicker one. Unfortunately, Theorem 9 does not provide an effective procedure for going from one program in the sequence to the next.

2. Let $r(x,y) = 2^x + 2^y$ and let f be the corresponding function of Theorem 9. Let g(x) be a lower bound on the time that it takes to compute f(x), that is, $g(x) < F_i(x)$ for every index i for f and for almost all x. The function g(x) can be taken to be at least as large as 2^x, since there corresponds to every index i for f an index j such that $F_i(x) > 2^x + 2^{F_j(x)}$, and therefore $F_i(x) > 2^x$ for almost all x. Given any such lower bound g(x), we know that $2^{2^{g(x)}}$ is also a lower bound, since there corresponds to the index j for f which is given above another index k such that $F_j(x) > 2^x + 2^{F_k(x)}$ and $F_k(x) > g(x)$ so that $F_i(x) > 2^x + 2^{2^x+2^{F_k(x)}} > 2^{2^{F_k(x)}} > 2^{2^{g(x)}}$ for almost all x. In this way we can obtain an increasing sequence of lower bounds on the computation time of f, starting with any lower bound that we choose. Note in particular that there can be no such thing as a greatest lower bound on the computation time of f.

3. Theorem 9 exhibits a function f, and in fact infinitely many distinct functions, so that the time required to compute f with any program can be halved for almost all x by some other program, that is, there corresponds to every index i for f another index j such that $F_i(x) > 2F_j(x)$ for almost all x. This suggests a possible way of dealing with the function

$$f(x) = \begin{cases} 1 & \text{if } 2^{2^x}+1 \text{ is prime} \\ 0 & \text{otherwise,} \end{cases}$$

which was introduced in section 2. The hypothesis there asserts that f = g, where

$$g(x) = \begin{cases} 1 & \text{if } x < 5 \\ 0 & \text{if } x \geq 5. \end{cases}$$

On the standard M-Computer, one can give a program for g whose computation time cannot be reduced by any other program. If one can show, therefore, that the computation time of every program for f can be halved for almost all x, or even for infinitely many x, then the hypothesis must be false.

4. Rabin has defined a partial ordering on the set of partial recursive functions in terms of the time required to compute these functions. Essentially, f is said to be more difficult to compute than g, written $f \succ g$, if there exists a program P_i for g such that $F_j(x) > G_i(x)$ for all programs P_j for f and for almost all x. Rabin's theorem, then, asserts that to every total recursive function g there corresponds a 0-1 valued total recursive function f such that $f \succ g$. We can now obtain another interesting fact. Let $r(x,y) = 2^y$ and let f be the corresponding total recursive function of Theorem 9. Let h and g be total recursive functions such that $h \succ f \succ g$. Then there exists a program P_i for g such that $F_j(x) > G_i(x)$ for all programs P_j for f and for almost all x. Theorem 9

asserts that there exists a program P_k for f such that $F_j(x) > 2^{F_k(x)}$ and, since $F_k(x) > G_i(x)$, it follows that $F_j(x) > 2^{G_i(x)}$ for almost all x. Similarly, one can show that $F_j(x) > 2^{2^{G_j(x)}}$ for almost all x, and so on. On the other hand, since $h \succ f$, there exists a program P_k for f such that $H_\ell(x) > F_k(x)$ for every program P_l for h. Since $H_\ell(x) > F_j(x) > 2^{F_k(x)}$, it follows that $H_\ell(x) > 2^{2^{F_k(x)}}$ for almost all x. This argument can be continued. Hence h is much more difficult than f and f is much more difficult than g, and since h and g are arbitrary total recursive functions satisfying $h \succ f \succ g$, it follows that no function comparable to f comes even close to being equal in difficulty to f.

M. Blum

References

1. Any effective list of all partial recursive functions is equivalent to the list ϕ_0, ϕ_1, ... determined by the standard M-Computer, as we show in Example 4.

2. The enumeration theorem asserts the existence of a partial recursive function f of two variables i and x such that

$$f(i,x) = \begin{cases} \phi_i(x) & \text{if } \phi_i(x) \text{ converges} \\ \text{divergent} & \text{otherwise.} \end{cases}$$

The s-m-n theorem, in a weak form that we require, asserts the existence of a total recursive function s such that $\phi_i(\langle x,y\rangle) = \phi_{s(i,x)}(y)$ for all i, x, and y.

3. M. O. Rabin, Degree of Difficulty of Computing a Function and a Partial Ordering of Recursive Sets, Hebrew University, Jerusalem, Israel, April, 1960.

4. The symbol $\langle\rangle$ denotes the standard coding of N × N into N: $\langle 0,0\rangle = 0$, $\langle 0,1\rangle = 1$, $\langle 1,0\rangle = 2$, $\langle 0,2\rangle = 3$, $\langle 1,1\rangle = 4$, $\langle 2,0\rangle = 5$, ...

5. A simplified form of Kleene's recursion theorem asserts that to every total recursive function s there corresponds an integer j such that $\phi_{s(j)} = \phi_j$.

6. H. Rogers, Jr., Recursive Functions and Effective Computability (McGraw-Hill Book Company, Inc., New York, in press).

C. OLFACTION

The slow potential of the frog's olfactory mucosa (the electro-olfactogram or EOG), which is recorded by a surface electrode when an odor reaches the nose, is a complex waveform. This has become apparent from a series of experiments in which a wide variety of odorous stimuli[1] was used. In addition to the large negative potential described by Ottoson,[2] there are at least two phenomena that occur before the negative swing and are generated at a different location from that of the negative potential. The early events are seen in different combinations with different stimuli and consist either of an initial positive swing (thought by Ottoson to be an artefact resulting from charged water-vapor molecules) or an initial small negative wave, or both. Figure XXVI-1 illustrates the

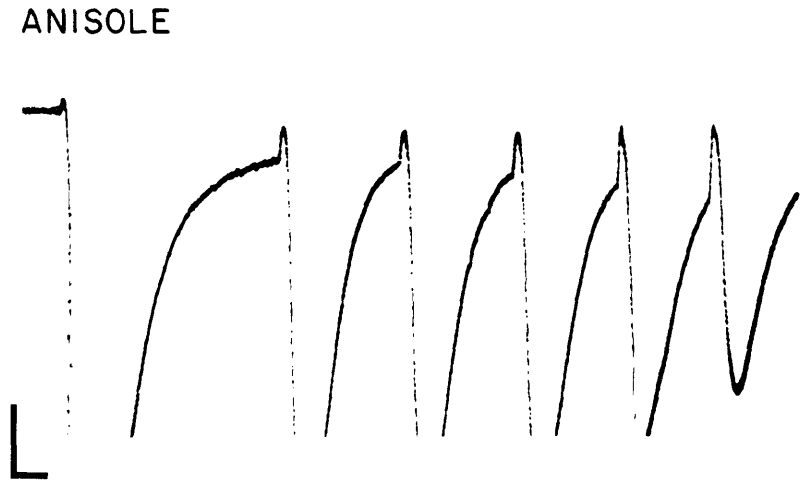


Fig. XXVI-1. The EOG recorded from the olfactory mucosa of the frog. Calibration marks in all figures indicate 1 mv (positive upward) and 1 sec. Short puffs of anisole vapor were delivered just before the sharp upward deflection. The response was recorded with a gelatin salt bridge electrode, 30 μ in diameter, touching the mucosa surface.

initial positive-going potential. It shows the response to 6 successive short puffs of anisole vapor on a single sweep 20 seconds in duration. The first puff elicits a small positive (upward) deflection followed by a large negative deflection. The next puff of odor appears to interrupt the negative process and return the potential very nearly to the resting value for a very short time before the large negative potential occurs again.

Following a notion recently advanced by Lettvin to account for membrane and receptor phenomena,[4] we suggest this explanation of the phenomena shown in Fig. XXVI-1: The odor molecules can have two effects on the membrane or receptor sites. Those stimuli to which a receptor or part of a receptor is sensitive in one way (that is, which cause an action potential to be generated) cause the receptor to depolarize there, which in these experiments shows up as a small initial negative potential. Other receptors that are sensitive in another way to the odor are actively clamped to membrane potential by a mechanism that causes a low impedance across the membrane without changing its potential. It is not unreasonable to connect the depolarization with sodium ion permeability, and the impedance shunt with potassium ion or chloride ion permeability. The tendency to return to resting potential independently of the magnitude of the large negative wave (shown in Fig. XXVI-1) is reminiscent of the crayfish stretch receptor.[5] There is no reason, in principle, why a stimulus, particularly a chemical one, cannot be inhibitory in the same sense as the postulated chemical transmitters. For a given stimulus there are many more nonresponding cells than there are responding units; hence the initial negative event is seen only under special conditions related to a highly sensitive

DICHLOROBENZALDEHYDE

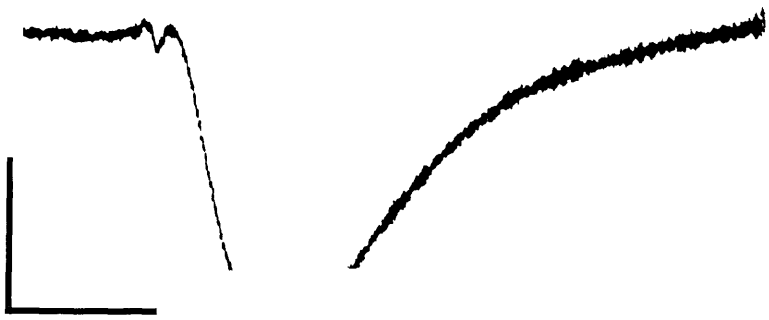


Fig. XXVI-2. Response to 2,4-dichlorobenzaldehyde.

GERANIOL

Fig. XXVI-3. Response to geraniol.

METHANOL

Fig. XXVI-4. Response to methanol.

depolarization effect and a rather insensitive shunt effect. This is shown in Fig. XXVI-2. Stimuli that can diffuse quickly and reach the receptor with very little spread in arrival time will show the initial events more clearly than slower-moving molecules. In the latter case, the action-potential response will mask most of the early events, as shown in Fig. XXVI-3. Methanol, which is odorless to the experimenters, produces the EOG shown in Fig. XXVI-4. It is positive only, with no sign of an initial depolarization and little or no sign of the action-potential negative wave.

R. C. Gesteland, J. Y. Lettvin, W. H. Pitts

References

1. R. C. Gesteland, Some Positive Aspects of Smell, Proc. Conferences on Recent Advances in Odor, New York Academy of Sciences, 1963 (in press).

2. D. Ottoson, Acta Physiol. Scand. (Stockholm) 35, Suppl. 122, 1956.

3. D. Ottoson, Acta Physiol. Scand. (Stockholm) 47, Suppl. 149, 1959.

4. J. Y. Lettvin, Quarterly Progress Report No. 70, Research Laboratory of Electronics, M.I.T., July 15, 1963, page 327.

5. C. Eyzaguirre and S. W. Kuffler, J. Gen. Physiol. 39, 87 (1955).