A. G. OETTINGER, Editor

Multi-Tape and Infinite-State Automata—A Survey

PATRICK C. FISCHER
Harvard University, Cambridge, Massachusetts

A survey of machines which are more powerful than finite automata and less powerful than general Turing machines is presented. It is felt that the machines in this category are as closely related to digital computers as either the finite automata or the unrestricted Turing machines.

Intermediate machines can be created by adjoining an infinite-state memory to a finite-state machine and then performing any or all of the following: (1) restrict the manner in which the unbounded portion of the memory can be accessed, (2) bound the number of steps allowed for a computation by some increasing recursive function of the length of the input, (3) restrict the total amount of memory available in the same manner. Examples from all three classes and their properties are discussed.

1. Introduction

In recent years there has been increased interest in abstract machines with more computing power than finite-state one-tape machines but with less potential complexity than Turing machines. Although a computer without auxiliary storage can be viewed as a finite-state device, and a computer with unlimited auxiliary storage can be regarded as a Turing machine (cf. Wang [65]), neither approach yields as much insight into the nature of computation as would be hoped.

To the computability theorist the regular sets of Kleene [36], which can be recognized by finite-state one-tape automata, are a very small subset of the recursive sets. The latter can be recognized by Turing machines [64] but, in general, not by machines of lesser computing ability. In between the regular sets and the recursive sets are uncountably many classes of sets of non-negative integers.[1] Some such classes are the deterministic pushdown languages [26], context-free languages [8], context-sensitive languages [8], elementary sets [28], primitive recursive sets [35], provably recursive sets [20] and many others.

Any one of the above intermediate classes may turn out to be more closely related to real computing than the regular sets or the recursive sets. Thus, machines with intermediate computing power may prove to be better models for real-world digital computers, at least for many purposes, than the better known finite-state and Turing machines.

In this paper an attempt is made to provide a census of nonprobabilistic intermediate machines. Discussion of finite-state machines and of unrestricted Turing machines is excluded, but familiarity with [51] and Part 1 of [13] is assumed. The terms *machine* and *automaton* are used interchangeably to mean any well-defined discrete system which defines a class of sets between the regular and recursive sets. The terms *acceptor*, *generator* and *transducer* designate respectively machines which have no output except "accept" or "reject," machines which have no input except a single starting pulse, and machines which have both inputs and outputs.

2. Multi-Tape Finite-State Automata

The simplest generalization of one-tape finite-state machines is the n-tape one-way acceptors introduced by Rabin and Scott. These devices recognize sets of n-tuples of strings via computations in which at each step one of the n input tapes is advanced one square and a new machine state is entered. The tape advanced depends upon the present state of the machine, and an n-tuple is accepted or rejected depending on whether the machine is in a final or nonfinal state after reading all of the input on all n tapes. Special end-of-tape symbols are permitted.

The deterministic and nondeterministic versions of these machines generate, for each n, classes of sets of n-tuples denoted by D_n and N_n, respectively. For $n > 1$, D_n is closed under set complementation, but Rabin and Scott showed it is not closed under union or intersection [51]. Rosenberg has shown that D_n is not closed under input reversal, concatenation or the closure (star) operator of Kleene [54].[2]

Unlike the case when $n = 1$, for $n > 1$, it does make a difference whether an n-tape automaton is deterministic or

This paper was presented as an invited talk at the 1964 International Colloquium on Algebraic Linguistics and Automata Theory, Jerusalem, Israel. Preparation of the manuscript was supported by National Science Foundation Grant GP-2880.

[1] One may often wish to work strings of symbols rather than non-negative integers. The usual procedure will be to associate with an integer the string representing its n-ary expression for some base n, generally $n = 2$.

[2] The reversal of string $\sigma_1\sigma_2 \cdots \sigma_{n-1}\sigma_n$ is the string $\sigma_n\sigma_{n-1} \cdots \sigma_2\sigma_1$ where the σ_i are each single symbols in the input alphabet. Reversal of an n-tuple is componentwise.

nondeterministic. An example by Rosenberg of a set in $N_2 - D_2$ is the set of all pairs of strings in which the second member of the pair is equal to a terminal segment of the first member (i.e., of the form $\langle xy, y \rangle$). Elgot and Mezei found that N_n is closed under union, concatenation, star and input reversal [15]. Since N_n is in fact the smallest class which contains all finite sets of n-tuples and is closed under union, concatenation and star, an n-dimensional analogue of Kleene's theorem of regular events holds for N_n. However, for $n > 1$, N_n is not closed under complementation or intersection.

In addition to being closed under fewer operations, the classes D_n and N_n are more complex than $D_1 = N_1$ (the regular sets) in several ways. Questions which are decidable for D_1 are often undecidable for D_n ($n > 1$). One such question is the intersection problem: to determine whether or not the intersection of two members of a class of sets is nonempty (cf. Rabin and Scott [51]). A question which is solvable for D_1 but open for D_n is to determine whether or not two machines in a class accept the same set of n-tuples of strings. The class D_2 contains properly the class of all relations definable by the "generalized sequential machines" of Ginsburg [25].

Ginsburg and Spanier have regarded multi-tape finite-state transducers as extended as generalized sequential machines [27]. For finite-state machines with two input tapes and one output tape they show that if the languages permitted on two of the three tapes are regular, then so is the third; if one is regular and one is context-free, the other is context-free; but if two are context-free, the third is not necessarily context-free. A generalization of this is given by Fischer and Rosenberg: for n-tape acceptors, if all but two of the n tapes are restricted to regular languages and one to context-free, then the language on the last tape will be context-free, etc. [24].

Elgot and Rutledge considered deterministic n-tape automata with tapes that are blank except for a single end symbol [17]. Unlike the general D_n case, the intersection question for the class of sets defined by these machines is solvable. This follows from the existence of an effective mapping of such sets into Presburger formulas (of the first order theory of addition of non-negative integers), for which the satisfiability question is solvable. If one permits more than one read head per tape, then one gets all of the Presburger relations. Also investigated were properties of the machines when one or more of the tapes are loops, rather than straight tapes. The question of whether a machine with two loop tapes (or one loop tape with two heads) accepts a nonempty set is unsolvable. For machines with one loop tape (with one head) and n straight tapes the intersection problem is open. It is solvable if Hilbert's Tenth Problem is.

3. Storage-Access Restricted Machines

Turning now to infinite-state devices, it is clear that in general one will have the computing power of a Turing machine. One can, however, obtain intermediate classes by imposing restrictions on the way a device can access its storage (i.e., on its state transitions) and/or by disallowing computations in which a given measure of complexity (e.g., the number of "steps") exceeds some constraint (usually a function of the input). The first of these approaches is discussed in this section.

The best known example of a storage-access restriction is the *pushdown* store, in which only the last item of information placed in the store is immediately accessible to the control unit of the computing device. (Pushdown stores are sometimes called LIFO stacks because of this last-in-first-out nature.) One-tape Turing machines in which the tape is used only as a pushdown store are called *pushdown-store machines*. Such machines are of interest to both automata theorists and mathematical linguists because of the important connection between the two areas given by Chomsky: a language is context-free if and only if it is recognizable by some nondeterministic pushdown-store acceptor [9]. Thus, results from linguistics can be used to prove theorems about pushdown-store automata, and conversely.

Another example of storage-access restriction is the *counter*. The value of a counter is any non-negative integer, and on a single machine step a counter's value can be increased by one, decreased by one, or tested for zero.[3] A counter can also be regarded as a pushdown store with a one-symbol alphabet.

Evey [18] and Fischer [21] investigated both deterministic and nondeterministic machines with one or more pushdown stores and/or counters. Relationships among acceptors, generators and transducers were explored and it was shown that a set which was accepted by a nondeterministic version of one of the machines under consideration could also be defined as the output of a deterministic transducer of the same type, and conversely. For example, the class of all sets which are the output of a deterministic one-counter machine, the class of all sets which are the output of a nondeterministic one-counter machine, and the class of all sets which are the domain (input) of a nondeterministic one-counter machine are all the class C' mentioned below.

By a result of Minsky [44], two-counter machines (and therefore two-pushdown-store machines) are already as powerful as Turing machines. Thus, the approach of Evey and Fischer yields exactly four distinct intermediate classes, which can be expressed as those sets recognizable by deterministic and nondeterministic one-counter acceptors (C and C') and those sets recognizable by deterministic and nondeterministic pushdown-store acceptors (P and P'). P' is, of course, the class of context-free languages and is closed under union, concatenation and star, but not under complementation or intersection (cf. Scheinberg [55] and Bar-Hillel, Perles and Shamir [5]). P is an interesting proper subclass of P' and is closed under

[3] The capabilities of the machines are unchanged if the counters are permitted to take on all integral values.

complementation, but not under union or intersection. The classes C and C' are not as interesting linguistically and have not received as much attention.

Early work with pushdown-store machines is described by Oettinger [47]. In [60] Schutzenberger exhibits a rather complicated algebraic definition of deterministic pushdown-store machines and proves some basic properties, from which the closure of P under complementation follows trivially. Recently, Ginsburg and Greibach have investigated P. With the exception of [26], most of their results have not yet been made public. P and P' and the associated machines have also recently been studied by Haines [29]. The fact that languages in P are always unambiguous is independently given in both [26] and [29].

Other kinds of storage-access-limited automata have been discussed in several papers by Schutzenberger. In [56] he deals with a class of finite-state transducers which are more powerful than generalized sequential machines and less powerful than nondeterministic, finite-state, one-input-one-output transducers. As in the other two cases, regular sets are preserved under such transductions.

In [57] acceptors with integer-valued counters are introduced. Unlike the counter machines previously mentioned, the state of the finite-memory part (control unit) of the machine is not permitted to depend upon the contents of the counters. (Thus, machines with two or more counters do not have the computing power of Turing machines.) For each combination of control unit state and input symbol, a machine performs a transformation on its counter space in such a way that the new value of each counter could be computed by a finite computer program (having sufficient temporary storage to handle intermediate results) using only the following kinds of operations: (1) addition or subtraction of two integers, (2) multiplication of an integer by a (bounded) integral constant, (3) reduction of an integer modulo a (bounded) integral constant. A machine accepts an input tape if the n-tuple representing the values of the counters after all input has been processed is *not* in a given subspace of the n-dimensional vector space over the integers, where the subspace selected depends upon the final state of the control unit of the machine.

Schutzenberger shows that his machines could be defined in terms of finite sets of integral $n \times n$ matrices, n depending on the machine. A machine is the homorphic representation of the monoid of inputs in the ring of matrices such that an input word is accepted if and only if the upper-right-hand entry in the matrix associated with the word is nonzero. Having demonstrated the equivalence, he shows by algebraic methods that the class defined by the entire family of his machines is closed under union, intersection, concatenation and star, but not under complementation. An analogue to Kleene's theorem is not produced, however, because the appropriate basis (which for the regular events is the class of finite sets) for the induction is unknown.

In [58] the above machines are further restricted by requiring that the ratio of the amount of information stored in the memory of the machine to the amount in the input tends to zero with the length of the input word. A smaller class results, which is closed under union, intersection and concatenation, but not under complementation or star. Results in [59] and [10] (with Chomsky) relate decision problems of several kinds of storage-access restricted automata, including pushdown-store machines, and of generative linguistic grammars to the appropriate algebraic structures. The notion of a formal power series is central to this development

4. Time Restricted Turing Machines

Some machines necessarily have their computation time bounded by a recursive function of the length of the input. For the finite automata of [51] the identity function suffices. On the other hand, for Turing machines there is, in general, no effective *a priori* bound on computation time, else the halting problem for Turing machines would be solvable.

The first person to consider time-restricted Turing machine computations was Yamada [66, 67]. Yamada was especially interested in a class of strictly increasing functions from the non-negative integers into the non-negative integers which were, in his terminology, *real-time countable*. Let the *characteristic sequence* of a set A of non-negative integers be the binary sequence $\alpha = \alpha_0\alpha_1\alpha_2 \ldots$ such that $\alpha_n = 1$ if and only if $n \in A$. Then a monotonic function f is real-time countable if some multi-tape Turing machine can generate the characteristic sequence of the range of f in real time, i.e., α_n can be generated by time n. The class of real-time countable functions was shown to be closed under addition, multiplication, composition, exponentiation and to contain all polynomials in one variable. Siegel found that there are real-time countable functions that are not primitive recursive, and monotonically increasing primitive recursive functions that are not real-time countable [62].

Hartmanis and Stearns studied time restrictions on Turing machines in a more general setting [31]. For any monotonically increasing function T, let S_T denote the set of all infinite binary sequences $\alpha = \alpha_0\alpha_1\alpha_2 \ldots$ such that there exists a multi-tape Turing machine which generates α_n by time $T(n)$ for all n. Such an S_T is called a *complexity class*. It is easy to show that for all monotone recursive functions, T, the complexity class S_T is a recursively enumerable set. Since every computable sequence is in S_U for some recursive U, it follows immediately that there are infinitely many complexity classes and that the class of all complexity classes cannot be effectively enumerated. The "real-time" case S_n (i.e., S_T where $T(n) = n$) is the minimal class.

Hartmanis and Stearns showed that multiplication of a timing function by a constant does not alter the complexity class, i.e., $S_{c \cdot T} = S_T$ for all monotone functions T and constants c. Complexity classes are also unchanged if more than one read-write head per tape is permitted. If

$\alpha \in S_T$ then there is a one-tape Turing machine which can generate α_n by time $(T(n))^2$, for all n. The same "square law" applies if one passes from multidimensional tapes to linear tapes.

Hennie and Stearns recently were able to show that the partial ordering of complexity classes has dense suborderings [34]. They first showed that if $\alpha \in S_T$ and T is real-time countable, then there is a two-tape Turing machine which can generated α_n by time $T(n) \cdot log(T(n))$. This permits the use of a diagonal method to show that if T is real-time countable and

$$\lim_{n\to\infty} \frac{T(n)\cdot \log (T(n))}{U(n)} = 0,$$

then S_U properly contains S_T. Since n and $[n^{1+\epsilon}]$ satisfy the conditions of the theorem, it follows that there are suborderings with the order type of the rationals.

Hennie also worked with time-limited recognition problems. In [33] he assumed that the input was placed on one of the working tapes of a Turing machine (where it could later be written over if desired). For this model he was able to show that the "square law" given above cannot be improved. For one-tape machines he also showed that if $2 > p > q > 1$ then S_{n^p} properly contains S_{n^q}. Although Hennie's "offline" acceptor is superfically only slightly different from the "online" version, where the input is placed on a special one-way tape which is not counted as one of the working tapes of the machine, the two models appear to have quite different properties.

The work of Hartmanis, Hennie and Stearns has raised many interesting questions. Is it true that if

$$\lim_{n\to\infty} \frac{T(n)}{U(n)} = 0,$$

then $S_T \neq S_U$? The binary sequence representing $\sqrt{2}$ is in S_{n^2}. Is it in any smaller class? Can one get relationships analogous to those in [18] and [21] covering generators, acceptors and transducers on the one hand, and online and offline machines on the other? For recognition, rather than generation, of sequences the equation $S_T = S_{c\cdot T}$ is replaced by $S_{T(n)} = S_{c(T(n)-n)+n}$ since reading of the input cannot be sped up. How does the special case $T(n) = n$ relate to the case $T(n) \geq (1+\epsilon)n$ for $\epsilon > 0$?

Rabin answered a question raised in McNaughton's 1961 survey of automata theory [42]: Can two-tape Turing machines do more in real time than one-tape machines? For the online acceptors, he showed that two tapes are better than one by requiring a machine to store two binary sequences and to retrieve either one of them upon demand, without delay [50]. One noteworthy aspect of this problem is that a proposition which at first appears to be quite simple requires a rather careful and involved proof. Furthermore, no one has yet been able to demonstrate for the same model that three tapes are better than two.

Blum has developed an abstract approach to computational complexity [7]. All he requires of a measure of complexity is that it be defined whenever the partial recursive function with which it is associated is defined and that one be able to tell effectively when a given complexity bound has already been exceeded by a computation (which may or may not subsequently terminate). Examples of acceptable measures are the number of steps in a Turing machine computation, the number of tape squares used in a computation, the maximum number of consecutive ones appearing on a tape during a computation, and so forth.

Blum shows that for any measure of computational complexity satisfying the above requirements and for any recursive function $g(x)$ there exists a recursive characteristic function $f(x)$, the computations of which can be "sped up" in the following sense. For any method of computing $f(x)$ there exists another method of computing $f(x)$ so that if $F(x)$ denotes the complexity measure of the first method and $F'(x)$ that of the second method, then $F(x) \geq g(F'(x))$ for all but a finite number of inputs x. (On the other hand, from the work of [31] and [34] it can be shown that there also exist functions which have complexity measure $F(x)$ but which cannot be "sped up" by $g(x) = x^{1+\epsilon}$.)

5. Tape-Restricted Turing Machines

The best known example of a tape-restricted machine is the linear-bounded automaton of Myhill [46]. A linear-bounded automaton is a one-tape Turing machine which is given only enough tape to hold the input string. Since the machine's alphabet may be larger than the input alphabet, this is equivalent to saying that the amount of information storage permitted is a linear function of the length of the input.

Kuroda has shown that the context-sensitive (Type 1) languages of Chomsky are exactly those sets which can be recognized by a nondeterministic linear-bounded automaton [39]. On the other hand, sets recognizable by deterministic linear-bounded automata are a Boolean algebra [39, 40]. Whether or not the deterministic and nondeterministic versions of linear-bounded automata define the same class of sets is still open. If the answer is yes, then the context-sensitive languages are obviously closed under complementation.

R. W. Ritchie discusses a hierarchy of functions based on tape-restricted computations [53]. He defines F_0 as the class of all functions computable by finite-state transducers. For each $i > 0$, F_i is then defined as the class of all functions which can be computed by a Turing machine where the amount of tape used is bounded by some function in F_{i-1}. A hierarchy of order-type ω is produced, and the union of all the F_i yields the elementary functions of Kalmár. The F_i are not closed under composition, but are closed under certain limited operations. If one begins with the linear-bounded automata instead of finite automata, a similar hierarchy is created, which interlaces with the first one and again yields the elementary functions.

In [37] the approach of [53] is relativized using functionals with tape-limited computations. In [38] Kreider and R. W. Ritchie consider the subclass of three-symbol deterministic linear-bounded automata. The subclass contains a "universal" machine but is not closed under substitution or identification of variables.

Very recently, Hartmanis, Lewis and Stearns have been working on tape-restricted computations [30]. The offline model of Hennie is modified to consist of a Turing machine with a two-way nonwriting input tape and a working tape. The amount of working tape used determines the complexity classification. Online machines are also being studied, and in either case the working tapes may or may not be restricted to pushdown-store operation. Hierarchies analogous to the one for time-limited complexity classes are given.

Despite the wording of the title of [30], as far as the writer knows, no one has yet explored in detail the interaction between tape and time restrictions. It would be nice if a reasonable trade-off of time for tape, and vice versa, existed so that a measure of computational complexity could take both into account in a meaningful manner.

6. Real-Time Iterative Arrays

A n-dimensional iterative array of finite-state machines consists of identical finite automata indexed by n-tuples over the non-negative integers [32]. Two machines are considered neighbors if their indices are identical in all but one component and differ by one in that position. Inputs and outputs for the array are connected to the machine indexed by (0,0,. . .,0). Except for the machine at the origin, the state of any machine in the array at time $t+1$ depends only upon the states of it and its neighbors at time t, and its state at time 0 is a particular quiescent state.

It can be shown that certain variations in the definition of an iterative array do not alter its properties. The origin machine may be allowed to be different from all the rest; the definition of neighbor may be relaxed somewhat; the initial states of the machines need not be quiescent so long as they are initially all the same.

It is easy to see that a one-dimensional array can simulate a Turing machine, since such an array can simulate any finite number of pushdown stores. Thus, other restrictions must be placed on iterative arrays to obtain recognition of intermediate classes of sets of strings. General time restrictions on iterative arrays, analogous to those of Hartmanis and Stearns for Turing machines, have not yet been investigated. However, the special case of real-time computation by deterministic iterative arrays of finite-state machines has been studied by several persons. The most comprehensive treatment is due to Cole [12].

One-dimensional iterative arrays can do many surprising things. Cole has shown that they can recognize in real time the set of all palindromes and the set of all strings of the form xx. The former set is, from a pushdown-store approach, inherently nondeterministic, and the latter set is not even context free. Fischer has shown that one-dimensional arrays can generate the characteristic sequence of the set of all prime numbers, i.e., the machine will put out a 1 at time t when t is a prime and a 0 at time t otherwise [22]. Atrubin showed that one-dimensional arrays can multiply in real time if the inputs are written in binary notation and fed in from right to left [1].

Cole showed that the class of sets recognizable by n-dimensional arrays is a Boolean algebra but is not closed under input reversal or concatenation. The set of all strings having a nonempty terminal segment of the form xx is shown not to be recognizable by any n-dimensional array. This set, however, is the concatenation of the universal set and the set of all xx, which are both acceptable by one-dimensional arrays. Furthermore, the reversal of this set is again recognizable by a one-dimensional array.

Cole's method can also be applied to the set of all strings having a terminal segment which is a nontrivial palindrome. This language is context-free but not acceptable by any n-dimensional array. Furthermore, for each n there exists a context-free language which is recognizable in real time by some $(n+1)$-dimensional array but not by any n-dimensional array. Thus, real-time iterative arrays and context-free languages are incomparable in a very strong sense.

7. Miscellany

Other work related to computational complexity is mentioned here briefly. Perles, Rabin and Shamir [49], Eggan [14] and Papert and McNaughton [48] have studied subclasses of the regular sets and the relationships between their linguistic and machine properties.

Lee [41] and Fischer [23] considered minimal sets of Turing machine operations and showed whether or not various restricted Turing machines could be universal. Shepherdson and Sturgis [61] and Elgot and Robinson [16] explored computations by machines having in place of tapes a finite number of registers, each of which can hold an arbitrary integer. Given the ability to make conditional transfers of control and to perform addition of two integers, such machines are equivalent to Turing machines.

Axt studied a hierarchy of recursive functions built up by beginning with the primitive recursive functions and creating new classes by uniform diagonal methods [2]. Related work by Fabian is given in [19]. Other work of Axt [3], Cleave [11], Meyer [43] and D. Ritchie [52] shows that several different methods of classifying the primitive recursive functions all produce hierarchies strongly related to that of Grzegorczyk [28].

An interesting survey paper by Bečvář may soon be available [6]. He gives a classification of the many models of computing devices used in investigations of computational complexity and offers some philisophical comment on the tenuousness of the relationship of some of them to "real" computing. Russian work in computational complexity, particularly that of Trahtenbrot [63], is well covered.

Finally, the author would like to mention that second-order bibliographical references will yield papers not cited above which are relevant to the area surveyed here, especially in the areas of mathematical linguistics and decision problems for automata. The bibliographies in [4, 6, 9, 10, 13, 35, 42, 45] are recommended to the reader seeking additional references.

RECEIVED JUNE, 1965

REFERENCES

1. ATRUBIN, A. J. An iterative one-dimensional real-time multiplier. Paper for Appl. Math., Harvard U., Cambridge, Mass., 1962. (ditto)
2. AXT, P. On a subrecursive hierarchy and primitive recursive degrees. *Trans. Amer. Math. Soc. 92* (1959), 85–105.
3. ——. Enumeration and the Grzegorczyk hierarchy. *Z. Math. Logik Grundlagen Math. 9* (1963), 53–65.
4. BAR-HILLEL, Y. *Language and Information.* John Wiley, New York, 1964.
5. ——, PERLES, M., AND SHAMIR, E. On formal properties of simple phase structure grammars. *Z. Phonetik, Sprachwiss. Kommunikationsforsch. 14* (1961), 143–172; also reprinted in [4].
6. BEČVÁŘ, J. Real-time and complexity problems in automata theory. Submitted for publication.
7. BLUM, M. A machine-independent theory of recursive functions. Doctoral thesis, Massachusetts Institute of Technology, Cambridge, Mass., 1964.
8. CHOMSKY, N. On certain formal properties of grammars. *Inform. Contr. 2* (1959), 137–167.
9. ——. Formal properties of grammars. *Handbook of Mathematical Psychology, Vol. 2.* John Wiley, New York, 1963, 323–418.
10. —— AND SCHUTZENBERGER, M. P. The algebraic theory of context-free languages. In *Computer Programming and Formal Systems*, North-Holland, Amsterdam, 1963, 118–161.
11. CLEAVE, J. P. A hierarchy of primitive recursive functions. *Z. Math. Logik Grunglagen Math. 9* (1963), 331–345.
12. COLE, S. N. Real-time computation by iterative arrays of finite-state machines. Doctoral thesis, Rep. BL-36, Harvard U., Cambridge, Mass., 1964.
13. DAVIS, M. *Computability and Unsolvability.* McGraw-Hill, New York, 1958.
14. EGGAN, L. C. Transition graphs and the star-height of regular events. *Mich. Math. J. 10* (1963), 385–397.
15. ELGOT, C. C., AND MEZEI, J. E. On finite relations defined by generalized automata. *IBM J. Res. Develop. 9* (1965), 47–68.
16. —— AND ROBINSON, A. Random-access stored-program machines, an approach to programming languages. *J. ACM 11* (1964), 365–399.
17. —— AND RUTLEDGE, J. D. RS-machines with almost blank tape. *J. ACM 11* (1964), 313–337.
18. EVEY, J. The theory and applications of pushdown-store machines. Doctoral thesis, Rep. NSF-10, Harvard U., Cambridge, Mass., 1963.
19. FABIAN, R. J. Hierarchies of general recursive functions and ordinal recursion. Tech. Rep., Case Institute of Technology, Cleveland, Ohio, 1964.
20. FISCHER, P. C. Theory of provable recursive functions. *Trans. Amer. Math. Soc. 117* (1965), 494–520.
21. ——. On computability by certain classes of restricted Turing machines. Proc. Fourth Ann. Symp. Switching Circuit Theory and Logical Design, Chicago, 1963, pp. 23–32.
22. ——. Generation of primes by a one-dimensional real-time iterative array. *J. ACM 12* (1965), 388–394.
23. ——. On formalisms for Turing machines. *J. ACM 12* (1965), 570–580.
24. —— AND ROSENBERG, A. L. Further results on nondeterministic *n*-tape automata. Abstr. 65T-48, *Amer. Math. Soc. Notices 12* (1965), 139–140.
25. GINSBURG, S. *An Introduction to Mathematical Machine Theory.* Addison-Wesley, Reading, Mass., 1962.
26. —— AND GREIBACH, S. Deterministic context-free languages. Abstr. 65T-155, *Amer. Math. Soc. Notices 12* (1965), 246.
27. —— AND SPANIER, E. H. Mappings of languages by two-tape devices. Proc. Fifth Ann. Symp. Switching Circuit Theory and Logical Design, Princeton, 1964, pp. 57–67.
28. GRZEGORCZYK, A. Some classes of recursive functions. In *Rozprawy Matematcyzne.* Warsaw, 1953, pp. 1–45.
29. HAINES, L. Generation and recognition of formal languages. Doctoral thesis, Massachusetts Institute of Technology, Cambridge, Mass., 1965.
30. HARTMANIS, J., LEWIS, P. M., AND STEARNS, R. E. Classifications of computations by time and memory requirements. Proc. IFIP Int. Congr. Vol. 1, 1965, pp. 31–36.
31. —— AND STEARNS, R. E. On the computational complexity of algorithms. *Trans. Amer. Math. Soc. 117* (1965), 285–306.
32. HENNIE, F. C. *Iterative Arrays of Logical Circuits.* MIT Press, Cambridge, Mass., 1961.
33. ——. One-tape off-line Turing machine computations. Submitted for publication.
34. —— AND STEARNS, R. E. Two-tape simulation of multitape Turing machines. Submitted for publication.
35. KLEENE, S. C. *Introduction to Metamathematics.* Van Nostrand, Princeton, N. J., 1952.
36. ——. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy (Eds.), *Automata Studies*, pp. 3–41, Princeton U. Press, Princeton, N. J., 1956.
37. KREIDER, D. L., AND RITCHIE, R. W. Predictably computable functionals and definition by recursion. *Z. Math. Logik Grundlagen Math. 10* (1964), 65–80.
38. —— AND ——. A universal two-way automaton. Submitted for publication.
39. KURODA, S.-Y. Classes of languages and linear bounded automata. *Informat. Contr. 7* (1964), 207–223.
40. LANDWEBER, P. S. Three theorems on phase structure grammars of Type I. *Informat. Contr. 6* (1963), 131–137.
41. LEE, C. Y. Categorizing automata by W-machine programs. *J. ACM 8* (1961), 384–399.
42. MCNAUGHTON, R. The theory of automata, a survey. In *Advances in Computers, Vol. 2.* Academic Press, New York, 1961, 379–421.
43. MEYER, A. M. Depth of nesting and the Grzegorczyk hierarchy. Abstr. 622-56, *Amer. Math. Soc. Notices 13* (1965), 342.
44. MINSKY, M. L. Recursive unsolvability of Post's problem of tag and other topics in theory of Turing machines. *Ann. Math. 74* (1961), 437–455.
45. MOORE, E. F. *Sequential Machines: Selected Papers.* Addison-Wesley, Reading, Mass., 1964.
46. MYHILL, J. Linear bounded automata. Wadd Tech. Note 60-165, Wright-Patterson AFB, Ohio, 1960.
47. OETTINGER, A. G. Automatic syntactic analysis and the pushdown store. Proc. Symp. Appl. Math., Vol. 12, Amer. Math. Soc. Providence, R. I., 1961.
48. PAPERT, S., AND MCNAUGHTON, R. On topological events. Submitted for publication.
49. PERLES, M., RABIN, M. O., AND SHAMIR, E. Theory of definite automata. *IEEE Trans. EC-12* (1963), 233–243.
50. RABIN, M. O. Real-time computation. *Israel J. Math. 1* (1963), 203–211.
51. —— AND SCOTT, D. Finite automata and their decision problems. *IBM J. Res. Develop. 3* (1959), 115–125.
52. RITCHIE, D. M. Complexity classification of primitive recursive functions by their machine programs. Abstr. 622-59, *Amer. Math. Soc. Notices 13* (1965), 343.
53. RITCHIE, R. W. Classes of predictably computable functions. *Trans. Amer. Math. Soc. 106* (1963), 139–173.

54. ROSENBERG, A. L. On n-tape finite-state acceptors. Proc. Fifth Ann. Symp. Switching Circuit Theory and Logical Design, Princeton, 1964, pp. 76–81.
55. SCHEINBERG, S. Note on the Boolean properties of context-free languages. *Inform. Contr. 3* (1960), 372–375.
56. SCHUTZENBERGER, M. P. A remark on finite transducers. *Inform. Contr. 4* (1961), 185–196.
57. ——. On the definition of a family of automata. *Inform. Contr. 4* (1961), 245–270.
58. ——. Finite counting automata. *Inform. Contr. 5* (1962), 91–107.
59. ——. Certain elementary families of automata. Proc. Symp. Mathematical Theory of Automata, Polytech. Inst. of Brooklyn, 1962, 139–154.
60. ——. On context-free languages and pushdown automata. *Inform. Contr. 6* (1963), 246–264.
61. SHEPHERDSON, J. C., AND STURGIS, H. E. Computability of recursive functions. *J. ACM 10* (1963), 217–255.
62. SIEGEL, J. Some theorems about Yamada's restricted class of recursive functions. IBM Res. Pap. RC-510, T. J. Watson Res. Ctr., Yorktown Hts., N. Y., 1961.
63. TRAHTENBROT, B. A. Turing computations with logarithmic delay. *Algebra i logika 3* (1964), 33–48 (in Russian).
64. TURING, A. M. On computable numbers, with an application to the Entscheidungsproblem. Proc. London Math. Soc., Ser. 2-42, 1936, 230–265.
65. WANG, H. A variant to Turing's theory of computing machines. *J. ACM 4* (1957), 63–92.
66. YAMADA, H. Counting by a class of growing automata. Doctoral thesis, U. of Pennsylvania, Philadelphia, Pa., 1960.
67. ——. Real-time computation and recursive functions not real-time computable. *IRE Trans. EC-11* (1962), 753–760.

LETTERS—continued from page 788

and Garwick [Re-Views of IFIP Congress 65. *Comm. ACM 8* (July 1965), 471].

First, I do not agree that the conference arrangements were beyond criticism. Regarding the ordering of the sessions, there were several unfortunate clashes, e.g., parallel sessions on programming (F2 and F3), parallel sessions on time-sharing systems (H3 and H4), parallel sessions on numerical analysis (H7 and H9). This is due to failure to find out beforehand any indication of what interested potential Congress members, as was done before the Munich Congress (IFIP 62). This lack of knowledge was also evident in the allocation of the rooms: I left a crowded meeting, packed into a smallish room, and went into a parallel session, with few people lost in a vast ballroom. Physical arrangements were also somewhat slapdash: blackboards were, in general, too small and all too often rooms with several pillars were used for sessions involving projection of slides. The folders that were supplied were useful and welcome, but why were they designed so that it was impossible to put one's name on them? Some note paper would have been invaluable.

I agree with Buxton's comments about the reading of papers whose full texts have been distributed. But why did we have the full texts *and* summaries (in four languages) of the invited lectures but no guidance about the contents of the submitted papers? Such abstracts would have helped us to get more out of the meetings—other congresses (including IFIP 62) manage to produce such abstracts, or even preprints.

Ancillary services: It is not their presence that is important but also when they are available. Certainly, there was a Post Office, which was closed for half the day and in fact was open mainly during afternoon sessions. An information counter (not official IFIP!) that was open only during sessions, with meal breaks coinciding with the breaks between sessions, and even then giving incorrect information (as I subsequently discovered), is of doubtful value.

Personally I was not perturbed by the shortcomings of the social program. I only remark that it is not impossible to find out beforehand how many people can be accommodated at the UN (and restrict the acceptances accordingly), nor is it impossible to find out several months ahead whether there is a concert or other performance on a particular night.

I enjoyed what I saw and heard at IFIP. However, I have the feeling that a little more attention to detail by the organizers would have made IFIP 65 a much more valuable experience for all participants.

PAUL A. SAMET
Computation Laboratory
The University
Southampton, England

Remarks on Computers and Aid to the Handicapped

Dear Editor:

I would like to offer a correction of Mr. Glaser's remarks: "On Computers and Aid to the Handicapped" [*Comm. ACM 8*, News (Oct. 1965), 638]. I am concerned here only with his comments about activities at New York University.

Mr. Glaser defines aphasic patients as ". . . those who have for one reason or another lost the power of speech." This definition is a misleading popularization. Aphasia is characterized by a reduction in vocabulary which is usually specific to particular word classes and with an associated reduced control of syntax. Aphasia is to be distinguished from other types of verbal impairment, in particular from verbal apraxia and dysarthria.

New York University has indeed embarked upon a program to augment patient treatment in aphasia using computer assisted therapy. However, actual patient-machine sessions are not scheduled to begin until sometime shortly after November 1, 1965.

We have had some experience in treating a single *anarthric* patient for a period of about six months. This was a cooperative exploration of our facilities and techniques conducted by ourselves and a technically trained patient of exceptional intelligence and broad experience. Regrettably, the patient was not "helped." On the contrary, it was found that the patient's needs were far in excess of our current capabilities. This project made clear the necessity for confining our present research in this area to clinical situations that fall within the scope of current sophistication.

G. ROSE
Section on Communication Sciences
New York University, New York