THE UNIVERSITY OF MICHIGAN
COLLEGE OF LITERATURE, SCIENCE, AND THE ARTS
Communication Sciences Department

Technical Report

REALIZATION AND COMPLEXITY OF COMMUTATIVE EVENTS

Richard Laing

ORA Projects 03105 and 08226

supported by:

DEPARTMENT OF THE NAVY
OFFICE OF NAVAL RESEARCH
CONTRACT NO. Nonr-1224(21)
WASHINGTON, D.C.

and

U.S. ARMY RESEARCH OFFICE (DURHAM)
CONTRACT NO. DA-31-124-ARO-D-483
DURHAM, NORTH CAROLINA

administered through:

OFFICE OF RESEARCH ADMINISTRATION ANN ARBOR

March 1967

RESEARCH PROGRESS REPORT

Title: "Realization and Complexity of Commutative Events," R. A. Laing, The University of Michigan Technical Report 03105-48-T.

Background: The Logic of Computers Group of the Communication Sciences Department of The University of Michigan is investigating the application of logic and mathematics to the theory of the design of computing automata.

Condensed Report Contents: A measure of complexity is defined for a class of expressions denoting behaviors of commutative machines. For an expression of complexity m there is an m tape machine which realizes in real-time the event denoted. The machine realizations can be of many kinds; the "simplest" realizations are those composed of finite counters of a single kind of input letter, combined with infinite state machines which check off numbers (mod p) of one kind of input letter against numbers (mod r) of a second kind of input letter. The state diagrams of the finite state machines are in the form of cycles (possibly with linear "lead-ins"); the state diagrams of the infinite state machines graphically are infinite cylinders (possibly with two-dimensional lead in arrays). For any machine with m counter tapes (as here defined), an m-complex expression can be obtained which denotes the event which is the behavior of the machine. Finally it is shown that for all m there are m-complex expressions which require m-counter tape machines for realizing in real-time the events denoted.

For further information: The complete report is available in the major Navy technical libraries and can be obtained from the Defense Documentation Center. A few copies are available for distribution by the author.

Realization and Complexity of Commutative Events

In the first section we indicate how from any commutative regular expression a tape machine realization may be obtained. (Most of these results have already appeared, albeit implicitly and piecemeal in [L-W] and [L], and will be presented here rather informally.) In addition we indicate how from any machine realizations of the sort considered above a commutative regular expression which denotes the machine behavior can be obtained. The second section of this paper considers the degree of complexity counter tape machines must attain in order to realize various classes of commutative regular events.

I would like to thank Donald Stanat of the Logic of Computers Group whose careful criticism of early drafts of this paper has led to the elimination of many of my rambling inaccuracies. That the present version of this paper is still so untidy is the result of my impatience and not of any failure on his part to point out my remaining lapses.

I

The use of regular expressions (RE) to denote the behavior of finite automata is now well known (see [K], [C-E-W], [M-Y]). Commutative regular expressions (CRE) are less well known. CRE are closely related to RE. If RE denotes a regular event (an event realizable by a finite automaton) then CRE denotes the event (set of sequences) consisting of the set of sequences denoted by RE and in addition all the sequences obtained by all possible permutations of the letter tokens of the sequences

of RE. Fundamentally then, CRE determine numbers of kinds of letters in a string; the position of the letters in the string is, essentially, of no consequence.

Despite the "weak" structure of these events, investigations of them have proven interesting and fruitful in several respects. The CRE constructed on an alphabet of a single letter always denote finite automaton behaviors, and these single letter, "naturally" commutative events have been employed to show that the (ordinary) language of regular expressions is not finitely axiomatizable (using only simple substitution as a rule of inference. This problem was posed implicitly in [K], and solved in [R]; for successful "finite" axiomatizations employing more powerful rules of substitution and inference, see [S] and [A].)

The algebraic properties of CRE have been considered in [L-W], and in [G], and recently in [P]; they have also been mentioned briefly in [B]; the relation of CRE to formal languages has been touched upon recently in [G-S].

As is noted in [L-W] and in [G-S], a CRE may denote the behavior of an infinite state automaton; i.e., it is not the case that for every CRE there is a RE denoting the same event. (For a simple proof, consider the event denoted by $C[(ab)^*]$ (C is the commutative closure operator; C[(ab)*] thus denotes the set of all strings in a,b, in which the number of a's is the same as the number of b's.); the intersection of this event with the (genuinely) regular event (a*b*) yields $a^n b^n$, which is known to be nonregular; but regular events are closed under the boolean operations; assuming C[(ab)*] is regular leads to a contradiction.)

In [L-W], a method is presented by which, given any CRE it can be decided whether or not the CRE denotes the behavior of a finite automaton; if the CRE does denote a finite automaton behavior, then there is an algorithm for finding the automaton. In [L-W], and in [L], informal techniques are developed for the construction of state diagram representations of minimal (finite or infinite) automata having as behavior any CRE event.

For the infinite state automata, the constructing of infinite counter-tape realizations (finite state automata to which are appended one or more infinite counter tapes) is discussed.

The CRE which denote the behaviors of infinite state automata, and their realizations by infinite counter machines are the principal consideration of this present paper. As will be seen below, the CRE and the events they denote are, intuitively, of varying complexity. In this paper, a natural ranking of the CRE is given, and this classification shown to parallel a natural ranking of machine realizations.

From [L-W], any RE has a related CRE (throughout this paper, unless otherwise specifically noted, all expressions will denote commutative events); the CRE can be put into a "quasi-normal form" consisting of a disjoint union of constituent CRE's (CCRE). (Every CCRE is also, of course, a CRE.) Each of the CCRE will consist of a concatenated sequence of words, some of which may be starred. For example (aab (aa)* (aab)*) might be a CCRE. These CCRE are always of star-height 1 (i.e., no star operator need ever act over another star operator since the commutativity property assumed always breaks down the effect of star-height). The "star-length" of a CCRE can however be indefinitely large. Star-length

is the number of concatenated starred words in a CCRE. Star-length of CCRE is closely bound up with the cardinality of the alphabet upon which the CRE are constructed. From [L-W] (in its general form, the result is by Give'on) any CRE over an alphabet of k letters can be written in the form of a union of constituent CRE in which the events denoted by the constituents are disjoint, and where the maximum star-length of any CCRE is k. This can be seen from a consideration of vector independence in k-space, each of the concatenated starred words determining a vector.

This suggests that a complexity ranking of CRE and their related events can be constructed on the (inter-related) measures of alphabet cardinality and star-length.

Definition: The alphabetic complexity of any unstarred word (regardless of number of kinds of letters) is 0. The alphabetic complexity of a starred word of a CCRE is k-1 where k is the number of different kinds of letters of which the starred word is constructed.

Definition: The expression complexity of a CCRE is the sum of the alphabetic complexities of the starred words of the CCRE.

Definition: The expression complexity of a CRE is the sum of the complexities of its components.

Definition: A CRE having complexity m will be called an m-CRE.

For example, the CCRE (aab)* has complexity 1 (and is thus a 1-CRE); the CCRE (abbc)* has complexity 2; the CCRE (aabb)* (abc)* has complexity 3; the CRE (aab)* (aabb)* (abc)* has complexity 4, (abb)* $\cup$ (aa) (aab)*

is a 2-CRE.

Since the alphabet cardinality and star-length of CRE are interdependent, the maximum complexity for CCRE on k letters is k(k-1) (thus the maximum complexity for CCRE on a one letter alphabet is 0, on a two letter alphabet is 2, on three is 6, four 12, etc.).

This ranking scheme gives complexity 0 to commutative expressions realizable by finite automata and complexity 1 to the CRE $(ab)^*$, (this is the "simplest" non-finite state commutative event). The scheme assigns $(abc)^*$ the complexity 2, while the "most complex" events on two letters (those with two concatenated starred words, each word in two letters, e.g. $(ab)^*(aab)^*$) also have expression complexity 2.

We wish now to show that this complexity ranking of CRE and CCRE has a counterpart in a complexity measure for the "simplest" machines realizing the CRE and CCRE. Potentially, there are many ranking schemes we might employ: number of states, number of feedback loops in logical network realizations, graphical-topological properties of state diagrams (accessibility, connectedness, planarity, etc.) speed of acceptance by linear bounded automata, tape direction reversals, number of tape sweeps, algebraic properties of the underlying group and semigroup structures, etc. The classificatory characteristic we employ here is number of tapes. Now since a (single tape) Turing machine can be constructed so as to compute any recursive function, and since CRE events are safely within the recursive sets, severe additional restrictions must be imposed on our tape machines before they begin to reflect, with much fineness of discrimination, the complexity differences among CRE. The first restriction we impose is

that our machines not be allowed to write on or to erase their tapes; that is we restrict ourselves to machines with infinite counter tapes (indefinitely extendible tapes, blank except for a single marker, called the zero, or "start" marker; these tapes can be viewed as counters capable of storing the positive and negative integers, and zero, the machine able to change the contents plus or minus one at each "atomic move", and capable of distinguishing only between blank squares standing for a positive or for a negative integer, and the marked square standing for the number zero).

(This restriction to counters is not a particularly strange choice for the events under consideration; the commutative property destroys most of the significance of order in the events, so that the problem of acceptance of elements of the events is reduced to distinguishing complex conditions for numbers of occurrences of letters.)

That this restriction to counters is in itself not very useful is seen clearly from [M] where it is shown that a machine with two (single-ended) counter-tapes can compute any recursive function. We must therefore impose an additional condition on our machines. This is the condition of "real-time" acceptance of events. We require that the machine signal its acceptance or rejection of an input sequence immediately upon reading and acting upon the final symbol of the sequence.

The machines we thus arrive at are finite state control automata to which are appended infinite counters, and which accept or reject in real-time.

Relating the capabilitess of these infinite state machines (taken together with finite state commutative machines) to commutative regular

expressions is the principal subject of this paper. We will show that any m-CRE can be realized as the union and intersection of behaviors of commutative finite state machines and at most m infinite counter-tape machines.

We begin by making more precise our notion of finite state commutative machine and infinite state commutative counter-tape machine.

Each CFSM (commutative finite state machine) is of the form C(k,p) where k is the length of a linear "lead-in" of states, and p is the length of a cycle of states. The CFSM can receive input sequences composed of a single letter and accept all and only sequences of length = $k + n \cdot p$, where k,p are the "lead-in" and cycle lengths as above, and n = 0, 1, 2, ... (That is the CFSM is a counter with index k and period p and accepts all sequences of length k mod p.)

A 1-CTM (a counter-tape machine with a single counter tape) receives input sequences constructed on a 2 letter alphabet. Each 1-CTM is an ordered quadruple $\langle \mathbb{C}_1, \mathbb{C}_2, CT, A \rangle$ consisting of $\mathbb{C}_1$, $\mathbb{C}_2$ each a cyclic counting transducer for one of the two kinds of input letters; CT, a counter-tape; and A, a finite set of acceptance conditions.

A $\mathbb{C}$ of a 1-CTM is completely specified by assigning values to the quadruple: $\mathbb{C} = \langle \sigma ; k,p; s \rangle$ where σ is one of the (2 kinds of) input letters; k,p, are index and period of the counter (as above for the CFSM and take positive integers or zero as values; and s is +1 or -1 (these are signals to move the counter-tape one square in a positive or negative direction). The σ and s of $\mathbb{C}_1$, $\mathbb{C}_2$ of a 1-CTM are constrained in that $\mathbb{C}_1$, $\mathbb{C}_2$ must have different σ, as well as having s of opposite sign.

The counter-tape CT is a (doubly) infinite tape, ruled into squares, but completely blank except for a single distinguished square, the "zero" square or empty counter symbol. The zero square is to be under scan initially.

The acceptance conditions A are of three sorts Z, P, N (for zero, positive, negative). The precise specification of acceptance conditions will require some preliminary definitions.

Consider the cycles of states p_1, p_2 (which are the periods of $\mathfrak{C}_1$, $\mathfrak{C}_2$ respectively) and consider the set of all pairs of states where the first member of the pair is a state of the p_1 cycle, and the second member of the pair is a state of the p_2 cycle. We partition the state pairs into three sets:

A) The singleton set consisting of the final state of the p_1 cycle, and the final state of the p_2 cycle.

B) The set of state pairs in which the ratio of the (partial) period achieved in p_1 to the (partial) period achieved in p_2 exceeds the ratio p_1 to p_2.

C) The set of state pairs in which the ratio of the (partial) period achieved in p_1, to the (partial) period achieved in p_2 is less than the ratio of p_1 to p_2.

Let us call these above three conditions state conditions A, B, C respectively.

Acceptance Conditions

Z : Z-acceptance in a 1-CTM takes place if the counter is at zero and state condition A obtains (otherwise non-acceptance).

P : P-acceptance in a 1-CTM takes place if the counter is at zero and state conditions A or B obtain, or the counter is positive (otherwise nonacceptance).

N : N-acceptance in a 1-CTM takes place if the counter is at zero and state conditions A or C obtain, or the counter is negative (otherwise nonacceptance).

(Notice that the acceptance characteristics for any given 1-CTM impose only a fixed and finite number of conditions. Notice also that we assume that knowledge of CT being to the right or to the left of the zero mark is always available, although we do not explicitly discuss how we might employ machine states to record this parity.

One way to handle the matter explicitly is to enlarge the definition of CT (the counter-tape) to include a tape-state monitor. When CT is zero the monitor observes this directly and registers zero; as soon as a +1 move signal is received, the monitor registers positive, and continues to register this until the zero count again appears on CT, at which the monitor registers zero; similarly for -1 and the negative region of the tape.

Yet another way in which the acceptance activity might be carried out is given in the example of machine construction in the final pages of [L].)

A 1-CTM operates in the following fashion. With each of the $\mathbb{C}_1$, $\mathbb{C}_2$ finite counters beginning in its initial state, and with the infinite counter CT set at zero, input sequences on two letters are received, each finite counter, undergoing transitions independently counting its

own kind of letter token. Every time a final state is reached in either of the finite counters an output move signal is sent to CT. (The $\mathbb{C}_1$, $\mathbb{C}_2$ can thus be viewed (taken together) as transducers from the input alphabet of two letters, to an output alphabet of +1, -1, 0 according as CT is to move right (positive), left (negative), or not at all.)

After receipt of each input symbol, the state pair of the finite counters together with the state of CT will determine acceptance or not of the input sequence so far received.

An m-CTM will consist of m 1-CTM operating synchronously, and will possibly include any number of CFSM (also operating in synchrony). That is, the behavior of a m-CTM is composed of unions and intersections of behaviors of constituent CFSMs and m 1-CTMs.

When an m-CTM is displayed as the synchronous behaviors of separate machines (CFSMs and 1-CTMs) combined by union and intersection symbols, we will call this the Decomposed Form of the m-CTM and may abbreviate it as an m-CTM-DF.

That every m-CRE can be realized by an m-CTM is the first result to be shown.

The description of the construction of m-CTM from m-CRE will be presented very informally. (The notions are genuinely very simple, but unfortunately any attempt to be formal produces a cloud of subscripts and superscripts, etc. which wholly obscures this simplicity.)

Let us exclude from our attention for the moment those events which are realizable by CFSMs (by results in [L-W] we can always distinguish these and obtain these required finite state zero-complexity automata).

We concentrate first on the constituent expressions of CRE. These constituent CRE have two parts, a prefix part (possibly absent) consisting of an unstarred word, and a body consisting of zero or more concatenated words, each of which is starred. If the CCRE is constructed on an alphabet of n letters, then the number of different kinds of letters in the prefix word or in any single word of the body is of course at most n. In addition the number of words in the body need be at most n (this is the star-length). The event denoted can be expressed as an (infinite) set of points in an n-dimensional integer lattice coordinate system. The body of the expression denotes an infinite oblique process of points in the lattice system; the prefix (if it is present) determines a relative origin at which the infinite oblique process denoted by the body is initiated.

We now make the following assertion: any constituent m-CRE event is realized by the intersection of the behaviors of a (possibly compound) finite state machine with the behaviors of zero or more 1-CTM, where the number of 1-CTM employed is m. That is, the m-CRE can be realized by a special form of m-CTM-DF.

We now briefly discuss further the prefix part of the CRE and explain why we are free to exclude consideration of it from much of what follows. The prefix denotes a finite set of conditions which must be satisfied along with the (infinite set of) conditions of the body of the expression. In the set of sequences in the event the expression prefix merely imposes the requirement that the ultimately accepted sequences contain at a minimum a certain fixed number of certain kinds of letters. In the inter-

pretation by n-dimensional coordinate system, the expression prefix merely translates, from the origin of the n-dimensional system to a relative origin, the initiation point of the infinite processes. Graphically, the prefix imposes "lead-in arrays" of states on machine state diagram realizations. The essential point is this: the conditions imposed by prefixes can always be realized by a single finite state machine a copy of which can be "prefaced" to each of the machines we require to realize the body of the expression. (The specific construction of these prefix, lead-in arrays is given in [L-W].)

We come now to the body of the expression. Each of the words of the body of the expression determines an oblique infinite process of points in n-space where n is the number of different kinds of letters in the word. Each of the words of the body can be viewed as a vector of points in n-space, all vectors having in common the origin point (or a relative origin point imposed by the prefix). The event denoted by the body, is given by all of the points determined by all sums of the vectors with non-negative coefficients. This therefore includes those which are in the periodic process on each of the vectors themselves, certain points which lie on the planes defined by each pair of vectors, and certain points which lie in the "volume" defined by the vectors (the volume bounded by the planes formed by vector pairs).

Let us call the volume defined by the vectors along with the planes formed by vector pairs, the region of the event.

The event forms, within the region, periodic patterns of points in the space. (The event does not necessarily include all the points in

the region.)

Imagine the pattern of points (denoting the event) which lies in the region to be extended throughout the whole n-dimensional integer lattice space. Let us call this set of points the Extension (abbrev. Ext) of the event. This set of points (the Ext of an event) itself denotes an event. From results in [L-W] and [L], the Ext event is always a finite automaton event, and the desired finite automaton is readily obtained. In particular if E is an event, then Ext(E) is the smallest finite state event which satisfies $E \subseteq \text{Ext}(E)$. (Consider the following example in two-space. Take as CCRE body the expression $E = w_1{}^*w_2{}^*$, each word being in two letters and, to avoid co-linearity, the ratios of the two kinds of letters different for w_1, w_2. E, by itself, denotes an oblique, fan shaped pattern of points in two-space (in the first quadrant). In the expression $\text{Ext}' = m(w_1) + n(w_2)$ let m,n run over the positive and negative integers, and zero. The "event" denoted, if expressed in two-space, extends the pattern of points denoted by $w_1{}^*w_2{}^*$ over the whole four quadrants. If we take only those words of Ext' which have positive value, we get Ext E, the Extension of $w_1{}^*w_2{}^*$ throughout the first quadrant. (Ext of an event as we are using it here might therefore be more accurately called Pos(itive) Ext.))

Lemma 1: a. Ext of a CCRE is a regular event.

b. Any Ext event is realizable by a finite state commutative machine (in the sense of [L-W]).

c. Any commutative finite automaton can be expressed as product of n finite counters (CFSM) (one counter for

each of the n orthogonal directions of the space).

(These results are contained implicitly or explicitly in [L-W].)

The Ext result gives us a machine which defines a pattern of points throughout our discrete descriptive space; in the region of the event (denoted by the body of the expression) the Ext event is precisely the event desired. The problem now is to show how (infinite) machines, having the desired complexity properties, can be employed to limit the acceptance space of the Ext event to the region of the event. That is we wish to intersect the behavior of our finite Ext machine with a k counter-tape machine (or with k 1-CTM) where the k-CTM will accept only in the region of the event. (From this point on it may be easier to think of the desired event as including all the lattice points in the region; this is of course, not always the case, but the functions of the infinite counters in realizing CCRE may be easier to understand if this nicety is temporarily ignored; the essential problem is seeing how counter-tape machines can be used to restrict acceptance behavior to the region of the event.)

Statement 1: The region is defined by the $\binom{n}{2}$ planes determined by the n "vectors" of the body of the CCRE.

Statement 2: Each of the "vectors" is completely determined by one of the separate words (of the CCRE "body"); (these words are each of the form $(a_1^p a_2^q a_3^r \ldots a_n^w)^*$.

Statement 3: Any word of the form $(a_1^p a_2^q a_3^r \ldots a_n^s)^*$ can be realized by an (n-1)-CTM.

Statements 1 and 2 are obvious. In Statement 3, the first 1-CTM checks p $\underline{a}_1$s off against q $\underline{a}_2$s, the second 1-CTM checks q $\underline{a}_2$s off against

r $\underline{a}_3$s, etc. Altogether (n-1) 1-CTM are required. An input sequence is in the event if and only if each of the separate n-1 tapes simultaneously read zero (and the n-1 control automata are in the proper internal states).

Lemma 2: The region of any event which has expression complexity m can be realized by an m-CTM.

Combining Lemmas 1 and 2 we get:

Theorem 1 (Construction): a. An m-CTM can be constructed to realize any m-CRE in real-time.

b. The maximum expression complexity for events in n letters can be realized in real-time by n(n-1) 1-CTM behaviors intersected with the behavior of a finite, single letter, cyclic counters (CFSM).

(Note that (despite (a) above) we have not provided a complete algorithm for the desired machines, although such an algorithm clearly exists. The obtaining of the prefix machines and the obtaining of the Ext machines can easily be made algorithmic. The obtaining of the desired 1-CTMs is also possible, although calculations would surely get laborious. Employing techniques of analytic geometry, take the vectors, pair by pair, obtain the plane defined, and find the intersection this plane will make with two of the coordinate system planes. Each of these two intersections of the plane of a vector pair with coordinate planes determines a line in the two respective coordinate planes. The total number of these lines in the coordinate planes will be n(n-1) where n is the number of vectors. These lines in the coordinate plane are each realizable by a 1-CTM; whenever the 2 1-CTMs determined by a pair of vectors are at zero count, the input sequence is denoting a point on the plane defined by the vector pair;

whether the next input letter leaves us in the plane defined by the vector pair, takes us further into the region of the event, or takes us out of the region, can be recorded internally. (The machines are assumed to be able to remember whether CT is at zero, is in its positive part, 0, or in its negative part.) Similarly for all other vector pairs, etc., the n(n-1) 1-CTM being sufficient to define the region of the event.)

The results outlined above, showing how, from any CRE a machine realizing the event denoted can be obtained, suggest the following converse problem: from any CTM, obtain a CRE such that the event denoted by the CRE is the behavior of the CTM.

Any m-CTM can be decomposed into CFSMs and 1-CTMs.

Lemma 3: From any m-CTM the (behaviorally) equivalent CTM-DF consisting of CFSMs and m 1-CTMs can be obtained.

This is a consequence of the definition of a m-CTM from constituent machines; an m-CTM is merely a congery of CFSMs and m 1-CTMs all acting in parallel, the m-CTM behavior being unions and intersections of behaviors of synchronously operating CFSMs and 1-CTMs.

Definition: If in the CRE language we allow the use of the intersection symbol $\cap$, we will call the resulting language CRE-$\cap$, and expressions in this language we call CRE-$\cap$ expressions.

Lemma 4: From any CTM-DF a CRE-$\cap$ expression can be obtained such that the event denoted by the CRE-$\cap$ expression is the behavior of the CTM-DF.

Sketch of Proof: A decomposed form CTM consists of a union and intersections of machine behaviors. The machines are of

two sorts, finite state cyclic counter automata, and single tape machines with very simple structure. CRE expressions for the finite state machines can be obtained by any of several simple procedures. The cyclic portion of the machine is expressed by a starred word, the word composed of letters all of the same kind, the number of them equal to the length of the cycle. In addition there may be a lead-in to the cycle. This lead-in is given by an unstarred word in the same alphabet as the starred word. Thus all CRE for the finite state machines will consist of expressions of the form w w_1*, w,w_1, being on the same alphabet (and each word possibly absent), e.g. aa(aaa)*. The 1-CTM expressions can also easily be obtained. They will be of the form w w_1*w_2* where w (possibly absent) is a word in at most two letters, w_1 is a word in two letters, the exact number of each kind of letter given by the lengths of the counting cycles within the machine (similar to the finite case), and w_2 (possibly absent) is a single letter. For example ab(aabbb)*b), b(aab)*, or (abb)*a* would be typical 1-CTM expressions. (See Appendices for further discussion.)

Replacing each of the separate machines by a CRE expression denoting its behavior, we arrive at a CRE which is a union of constituent expressions, each of which constituents may contain intersections of separate CRE expressions. Now while union, dot, and star are in the CRE

language, intersection is not included in the CRE language (the original RE language from which the CRE language is derived does not contain intersection). Thus we arrive at a CRE-$\cap$ expression, not a CRE expression.

This restriction to the language with intersection can be removed. From results in [L-W] and in [E] the intersection of the events denoted by any two CRE can be found, and a CRE expressions for the intersection event obtained. (That is, taking care to include the empty set, etc., in the formulation, CRE events form a boolean algebra.)

Lemma 5: [L-W] For any CRE-$\cap$ expression an equivalent CRE expression can be found.

The question remains whether the conversion to CRE expressions of members of the particular class of CRE-$\cap$ which are produced from CTM-DFs preserves complexities.

We must consider CFSMs and 1-CTMs and their corresponding expressions and events. Recalling from the sketch of proof of Lemma 4 above, the CFSM expressions are of the form $w\ w_1^*$ (w, w_1 in the same letter) while 1-CTM expressions are of the form $w\ w_1^* w_2^*$ (w in at most two letters, w_1 in two letters, w_2 a single letter). Unions of the events cause no difficulty since these have expressions which (owing to the form of the complexity criterion) are of complexity at most that of the sums of the constituent parts. We must therefore examine the expression complexities which result from intersections of CFSM and of 1-CTM events. Intersection of two CFSM events is again a CFSM event, and thus are always expressible in 0-CRE form. The source of additional complexity (if any) must therefore lie in

intersections involving 1-CTM events.

The intersection of a 1-CTM event with a CFSM event can be empty, a finite set of sequences, or an infinite set of sequences. The first two kinds of intersection are expressible by 0-CRE. Intersections of 1-CTM events and CFSM events which are infinite sets of sequences are themselves at most 1-CTM events. Thus intersections involving 1-CTMs and CFSMs do not increase complexity of the CRE which denote these intersections. The intersection of two 1-CTM events can be empty, a finite set of sequences, or an infinite set of sequences. Again only the last case is a potential source of difficulty. The CRE for the (infinite set) intersections can be 1-CTM events or an event realizable by a CFSM and at most two 1-CTMs. Each of the two 1-CTMs has 1-CRE complexity, so again the degree of expression complexity is not increased.

Lemma 6: For any m-CTM-DF an m-CRE which denotes the event which is the behavior of the m-CTM-DF can be obtained.

Combining Lemma 6 with Lemma 3:

Theorem 2: (Machine Analysis and Complexity Preservation): For any m-CTM an m-CRE which denotes the event which is the behavior of the m-CTM can be obtained.

Note that 1-CTMs, CFSMs and synchronous combinations of them are the only form of real-time accepting multi-tape commutative machines we have defined with sufficient precision to warrant speaking of procedures which from a machine produce a commutative regular expression. It is possible that an alternative, more natural definition of multi-tape commutative machines can be given, the behavior of this class of machines

being precisely the class of commutative regular expressions. Among the characteristics which the 1-CTMs and CFSMs (and synchronous combinations of them) possess and which probably would also be required of any alternative counter-tape machine formulation are:

1. A particular kind of input letter is assigned a particular counter-tape move direction, and this correlation is never changed in the course of the computation (as a function, for example, of receipt of a special input-sequence symbol).
2. No auxiliary marking on the tape is ever permitted.
3. Although acceptance conditions are a function of (among other things) the counter tape being in a positive, a negative, or the zero configuration, no internal state transitions or tape directional moves are ever a function of tape configuration.

II

In this paper so far, we have indicated how, from any commutative regular expression with complexity m, we can obtain an automaton, with m counter tapes, which will realize in real-time, the event denoted by the expression. Conversely we have indicated how from any commutative automaton (in a particular form) having m tapes, a commutative regular expression with complexity m can be obtained, the expression denoting an event which is the automaton behavior. We have as yet said very little about the conditions under which a certain number of counter tapes will not only suffice but is absolutely necessary to attain real-time realizations. It is this latter problem, that of <u>required</u> number of tapes

(which provides a measure of underlying event complexity rather than merely expression complexity) to which we now turn. Let us first review some of what is known of machine requirements for realizing commutative regular events. First, events (commutative regular events) on an alphabet of a single letter are all finite automaton events. Events on alphabets of more than a single letter may not be finite automaton events. As we have mentioned, results of Wright [L-W] provide a decision procedure for distinguishing the finite from the infinite automaton events, and also obtaining the desired finite automata. Such a decision procedure is valuable, since a commutative regular expression may appear to be an infinite automaton event (and could even have arbitrary high expression complexity) and yet be finite automaton realizable. The reason for this curious state of affair will be seen by the following: geometrically speaking, the finite commutative regular events, if expressed in n-space (n being the cardinality of the alphabet upon which the events are constructed) form an orthogonal pattern of points (that is, all of the points in the event can be generated by perpendicular vectors) throughout the space of the event; the infinite automaton events form oblique processes of points in the space. It is entirely possible for 1) an infinite automaton event to be included in a finite automaton event, or 2) a union of infinite automaton events to form a finite automaton event. For example, in 2-space (e.g. the first quadrant of an ordinary x,y coordinate lattice) the event corresponding to all the points lying on or below the main diagonal is an infinite automaton event; a second event corresponding to all the points lying on or above the main diagonal is also an infinite

automaton event; the union of these two events is the trivial commutative finite automaton accepting any sequence whatsoever.

The principal result of this section will be to show that:

Theorem 3; Where k is the alphabet cardinality, there are expressions consisting of single words starred, which denote events of complexity k-1, for all k.

Theorem 4: Where k is the alphabet cardinality, there are expressions, consisting of concatenations of single words starred, which denote events of maximum complexity k(k-1), for all k.

As an approach to these general results let us first prove the following parallel related special results:

Theorem 3': a) The event denoted by the CRE (ab)* is realized by a 1-CTM and not realizable by a finite state automaton.

b) The event denoted by the CRE (abc)* is realizable by a 2-CTM and not realizable by a 1-CTM.

Theorem 4': Any event denoted by CRE of the form $w_1^*w_2^*$ (w_1, w_2 in two letters each, and non-collinear) is realizable by a 2-CTM (or by 2 1-CTMs) and not realizable by a 1-CTM.

Notice that all of the above assertions agree with the expression complexity requirements discussed in the first section of this paper.

The event of 3'a) is already known to be non-regular; this was discussed early in Section I. Therefore a 1-CTM is obviously not only sufficient but necessary. The decision procedure for distinguishing between finite and infinite automaton events, which in this case gives us our sought after minimal number of counters is not readily applicable in

sorting out degrees of infinite machine complexity however. Let us therefore, with 3'a) and the other events, establish some numerical measures of event complexity.

Where t is the length of an input sequence (and also the number of atomic moves allowed any realizing tape machine) how many essentially different sequences relative to the event under discussion, must be kept distinct (stored separately, coded differently) in order that the realizing machine be capable of operating properly?

Two finite sequences are <u>essentially different</u> relative to the event E if there exists a finite continuation sequence which if applied to each of the two sequences results in combined sequences leading in one case to acceptance, and in the other case to rejection, (i.e., the two original sequences can not be put in the same equivalence class vis-a-vis eventual acceptance or rejection). The relation "essentially different" thus is a right congruence which induces an equivalence over the set of all sequences in a natural way.

<u>Lemma (to 3'a)</u>: There are at least $2t+1$ classes of essentially different sequences relative to the event $(ab)^*$ of length $\leq t$.

<u>Lemma (to 3'b)</u>: There are at least $\frac{3t^2+3t+2}{2}$ classes of essentially different sequences relative to the event $(abc)^*$ of length $\leq t$.

The first of these above lemmas can be seen as merely insisting that there must be at least as many equivalence classes as can be indexed by employing the following input sequence representatives: Λ, a, b, aa, bb,

aaa, bbb, ... etc., the maximum different classes possible for sequences $\leq$ t in length being (2t+1). In effect all pairs consisting of the two letters, a and b (whenever occuring and in any order) may be deleted, only the excess of a's or of b's being recorded. Similarly in the second lemma any abc triples can always be deleted, the resulting number of classes being the number of sets of t or fewer objects drawn from three kinds of objects chosen so as always to exclude one kind of object.

Lemma (to 4'): There are at least $\frac{t^2+3t+2}{2}$ classes of essentially different sequences relative to events of the form $w_1{}^*w_2{}^*$ of length $\leq$ t.

In this lemma, a preliminary result (by Wright in [L-W]) may be useful. Events of the sort considered here have as state diagram realizations the whole first quadrant integer lattice (with points as states, the origin being the initial state). No states (points) can be merged. (If any two first quadrant points are merged, then a continuation sequence can always be found such that, (although a single state will result), from one of the original states a final state should have been reached, and from the other a non-final state should have been reached.) Returning to the complexity of the event of the third lemma, the input sequence of no length leaves us at the initial state (origin), an input sequence one letter long can take us to one of the two states, (0,1) or (1,0), two long to one of three different states, etc. For t = 0, 1, 2, etc. this is the sum 1 + 2 + 3 + 4 ... . For n = 1, 2, 3, ... , the formula for 1 + 2 + 3 ... is $\frac{n(n-1)}{2}$. Substituting t+1 for n we get the $\frac{t^2+3t+2}{2}$ of the lemma as the minimum number of right congruence classes.

These event complexities of the above lemmas must now be related to machine capacities. Any finite automaton with a fixed number of states h can distinguish at most h "essentially different" equivalence classes of input sequences. Since in the events considered here the number of equivalence classes increases indefinitely, a fixed, finite automaton can not realize the events in question. If an infinite counter is appended to the machine, then (for a single counter) by time t the machine can register (by being in a different tape-count — machine configuration) at most $h(2t+1)$ distinctions. For a 2-CTM the distinguishing capacity is $h(2t+1)^2$; similarly a machine with m counters, can by time t have distinguished at most $h(2t+1)^m$ different equivalence classes of input sequences.

If a machine, by time t, can not achieve as many tape — read-head configurations as there are classes of essentially different sequences, then there are at least two essentially different sequences which will receive the same coding-configuration. But in that case, a "falsifying" continuation input sequence can always be found, and the machine will fail to operate properly.

By comparing the machine capacities to the event requirements we see that the special results are proven (h is fixed and t = 0, 1, 2, ...).

3'a): $h < (2t+1) < h(2t+1)$

3'b): $h(2t+1) < \frac{3t^2+3t+2}{2} < h(2t+1)^2$

4': $h(2t+1) < \frac{t^2+3t+2}{2} < h(2t+1)^2$

The above special results in 3' suggest that not only can any expression consisting of a single word starred always be realized by a m-CTM

(where m = k-1, k = cardinality of alphabet on which the word is constructed) but that there are always events denoted by such expressions which require m-CTM for real-time realization.

Lemma (to Theorem 3): The number of classes of essentially different input sequences relative to events denoted by single words starred is given by a (k-1)th order equation in t (the length of the input sequences) where k is the alphabet cardinality of the word.

In particular, let us consider single words of the form a_1, a_1a_2, $a_1a_2a_3$, ... (that is, words with single occurrences of each of k letters, for all k). The length ℓ of an input sequence x is $\ell(x)$, $\ell(x) = t$. If P(k,t) is the number of ways of choosing t things from k things (repetitions allowed) in such a way that not all the k things appear, (this insures the "essential difference" of the sequences) then we want to show that $\sum_{t=1}^{T} P(k,t)$ is a polynomial of degree k-1 in T. It is sufficient to show that P(k,t) is a polynomial of degree k-2 in t, since $\sum_{t=1}^{T} t^n$ is a polynomial of degree n+1 in T.

But P(k,t) is just k times $\binom{k-1 + t-1}{t}$, the number of ways of choosing t things from k-1 with repetitions, since we can delete one of the k things and choose t to ensure that not all k appear, and do this in k different ways.

$$\text{So } P(k,t) = k\binom{k + t - 2}{t}$$

$$= k\frac{(k + t - 2)!}{(k-2)!t!}$$

$$= \frac{k\cdot(k + t - 2)(k + t - 3) \ldots (t + 1)\cdot t!}{(k - 2)!t!}$$

$$= \frac{k}{(n-2)!}(k + t - 2)(k + t - 3) \ldots (t + 1)$$

a polynomial of degree k-2, as required.[1]

---

[1]This form of the proof was suggested by Stewart Bainbridge of the Logic of Computers Group.

Comparing this result with the distinguishing capacities of counter tape machines, we have proved Theorem 3. Note that the results of this lemma apply to starred words having more than a single letter of each kind, the kinds not the number of each kind of letter determining the complexity. The (k-1)-CTM required are easily obtained.

We now begin consideration of Theorem 4.

When a starred word is concatenated with one or more starred words (excluding cases of co-linearity, etc.) the complexity of the resulting event increases (up to k words concatenated, k being alphabet cardinality, after which additional words need not increase the event complexity).

1. Each of the words separately imposes a t^{k-1} order polynomial number of essentially different input sequences which must be kept distinct.
2. The maximum $k \cdot t^{k-1}$ order polynomials distinctions can be realized by a k(k-1) CTM (from the construction results of Part I).
3. Concatenation does not decrease the number of distinctions required.

This gives us Theorem 4.

Let us briefly summarize by noting that the above results mean that there are always CRE of complexity m, which require CTM with m infinite counters for realization in real-time; the CTM can be given very simple

form: a union of behaviors of constituent machines, each of which constituent machine behavior is the intersection of behaviors of some extremely simple kinds of machines:

1) Infinite counter machines, the algebraic structure of the machine transition system being that of an infinite (commutative) semigroup with a sub-semigroup that is the product of the infinite additive group of integers $\langle C,+\rangle$ with $\langle C_n,+\rangle$, the integers mod n.
2) Finite machines in a single letter input alphabet, the algebraic structure being a (commutative) semigroup with a cyclic subgroup C_n.

Graphically, the state diagrams of the transition structures of the infinite machines are infinite cylinders (the subgroup), to which "one-way lead in" arrays of states are attached (the part of the semigroup which lacks inverses), while the state diagrams of the transition structure of the finite machines are finite loops of states (the cyclic group) with "pre-tails" of states (the essentially "one-way" semigroup portion).

(Some of these matters are discussed in [L-W].)

Let us conclude by pointing out several of the many problems remaining:

1. Find a more natural definition for the class of counter-tape machines which realize all and only commutative regular events.
2. Obtain precise Synthesis and Analysis algorithms (not merely sketches of intuitively adequate procedures, as has been done in this paper) relating CRE and CTM.
3. Obtain an algorithm which from any CRE would produce the tape-minimal CTM realization. (This problem might be attacked by employing generalizations of the decision procedure (of Wright in

[L-W]) for distinguishing finite state automaton events from infinite state automaton events. If the CRE can be suitably decomposed, perhaps multiple testing (by means of the decision procedure) of combinations of events denoted by the decomposed CRE would permit the "maximum" grouping of the finite state machine realizable portions on the one hand and the "most compact" grouping of the infinite state machine portions on the other.)

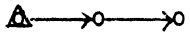
APPENDIX: A Sketch of the Construction of a 2-CTM Given a 2-CRE

1. 2-CRE = $(aabb)\cdot(abbb)^*\cdot(aab)^*$

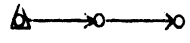
Allowing intersection and the Ext operation, 1. can be re-expressed as:

2. $[(aabb)]\cdot[Ext[(abb)^*\cdot(aab)^*]\cap[(\Lambda \cup ab \cup abb)(abbb)^*a^*]\cap[(\Lambda \cup ab)(aab)^*b^*]$

3. The prefix (aabb) is realized by: $\mathbb{C}_a(2,0) \times \mathbb{C}_b(2,0) =$

a-M: 

x

b-M: 

(In the above a-M is the a input machine, b-M the b input machine. The product of these is the desired machine. In the state diagrams above and later o is an ordinary state,  the initial state, a final state, and  a state which is both initial and final.)

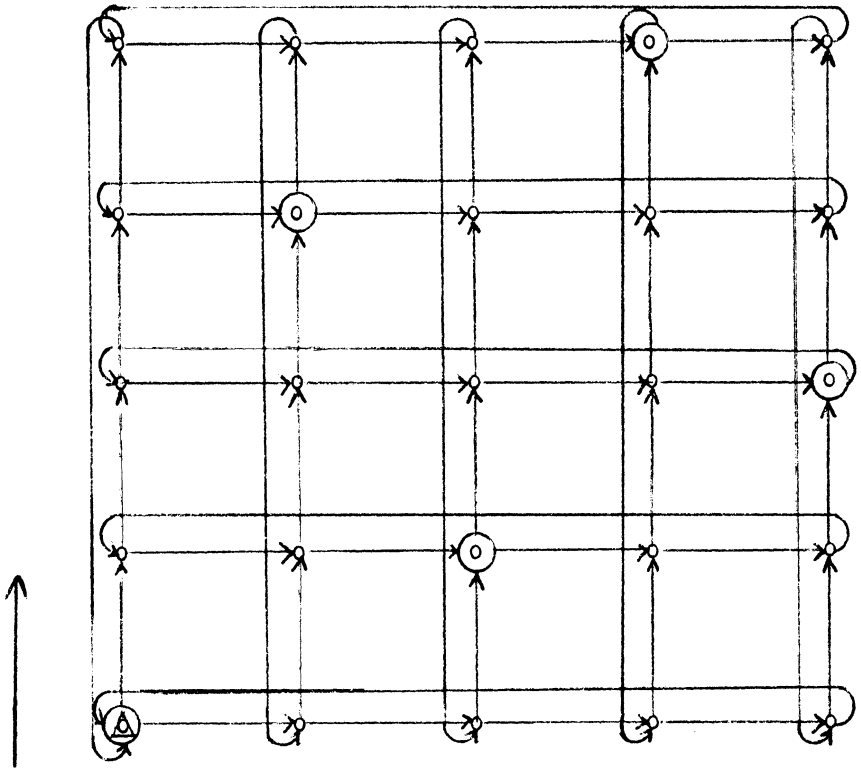
This "prefix" machine will "preface" each of the machine constituents.

4. Construction of the Ext machine:

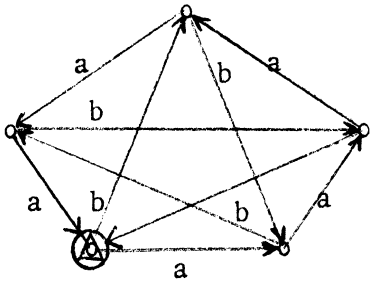
$Ext[(abbb)^*\cdot(aab)^*] = [(\Lambda \cup (aab) \cup (abb) \cup (aaaabb) \cup (aaabbbb))$

$\cdot(aaaaa)^*\cdot(bbbbb)^*]$

From methods in [L-W] and [L], an orthogonal finite state machine for this event is:



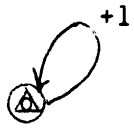
The minimal finite state automaton for the Ext event is:

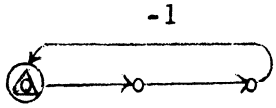


The complete decomposed form of the Ext machine (the separate CFSM) is given under step 6. (M_1 through M_5) below.

5. Construction of a 1-CTM for $(\Lambda \cup ab \cup abb)\cdot(abbb)^*a^*$

The 1-CTM will have three parts: an a-cycle part, a b-cycle part, and an infinite counter along with counter acceptance conditions.

1) a-M: 

2) b-M: 

3) A:P

The above diagram is to be interpreted as follows: the machine receives input sequences composed of a's and b's; the a-cycle part of the machine counts single occurrences of a in the input and for each a sends a move right (positive) instruction to the counter. The b-cycle counts b's by three in the input, and for each three sends a move left (negative) instruction to the counter. The a-M and b-M parts of the machine may be viewed as a transducer which maps sequences of a's and b's into sequences of 0, 1, and -1. These latter sequences are inputs to the infinite counter of the machine.

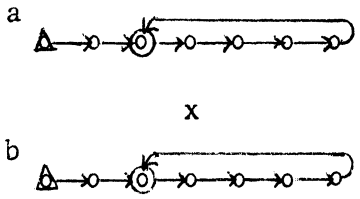
For the machine example above A:P is the acceptance condition. 1-CTM will signal acceptance only when its tape is in a positive count or when in zero count its $\mathbb{C}_1$, $\mathbb{C}_2$ state pair is also "positive".

All of the completed constructions are given below.

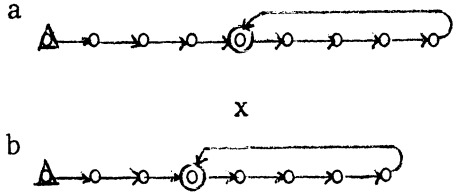
6. Summary

$2\text{-CRE} = \text{CRE} - \cap = [E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5] \cap [E_6 \cup E_7]$

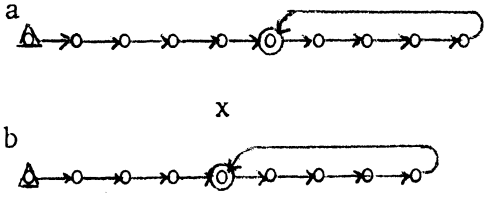
$E_1 = aabb(aaaaa)^* \cdot (bbbbb)^* \Longrightarrow M_1 =$



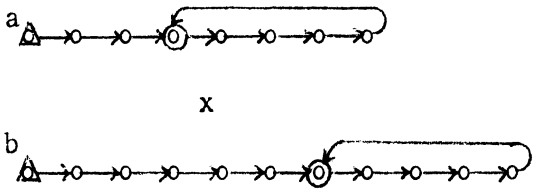
$E_2 = aabb(aab) \cdot (aaaaa)^* \cdot (bbbbb)^* \Longrightarrow M_2 =$



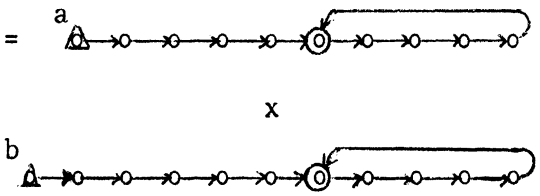
$E_3 = aabb(aaabb) \cdot (aaaaa)^* \cdot (bbbbb)^* \Longrightarrow M_3 =$



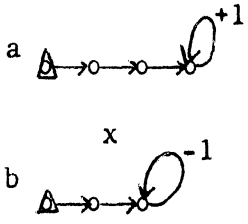
$E_4 = aabb(abbbb) \cdot (aaaaa)^* \cdot (bbbbb)^* \Longrightarrow M_4 =$



$E_5 = aabb(aaabbbb) \cdot (aaaaa)^* \cdot (bbbbb)^* \Longrightarrow M_5 =$



$E_6 = aabb(\Lambda \cup ab) \cdot (aab)^* b^* \Longrightarrow M_6 =$



A:N

$E_7 = aabb(\Lambda \cup ab \cup abb)\cdot(abbb)^*a^* \Longrightarrow M_7 =$ a-M: 

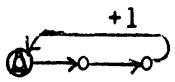
x

b-M: 

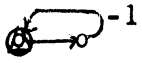
A;P

APPENDIX 2: A Sketch of the Method of Obtaining a 2-CRE Given a 2-CTM (in Decomposed Form)

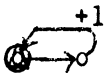
1. Given:

M_1 = a-M: 

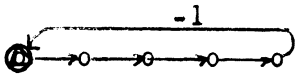
x

b-M: 

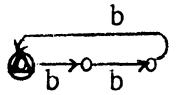
A:P

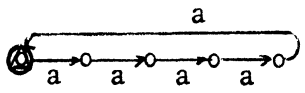
M_2 = a-M: 

x

c-M: 

A:Z

M_3: 



$B(M_1) \cup B(M_2) \cup B(M_3) = B(2\text{-CTM})$

($B(M_1)$ is the <u>behavior</u> of machine M_1, etc.)

The desired expressions can be read-off directly.

$B(M_1) = (\Lambda \cup aab)\cdot(aaabb)^*a^*$

$B(M_2) = (aaccccc)^*$

$B(M_3) = a(aaaa)^*\cdot(bbb)^*$

$2\text{-CRE} = (\Lambda \cup aab)\cdot(aaabb)^*a^* \cup (aaccccc)^* \cup a(aaaa)^*\cdot(bbb)^*$

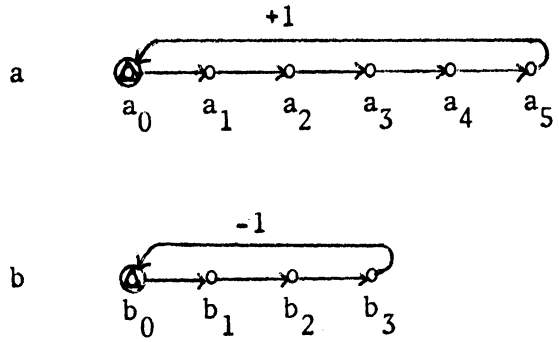
APPENDIX 3: Construction of a CTM for Expressions of the Form $(w_1 \cup w_2 \cup \ldots \cup w_n)w^*$

For simplicity in exposition we have largely restricted our discussion of constructions to the "basic" case of a starred word having a prefix consisting of a single word. In the case of multiple prefix expressions, the degree of expression complexity can be arrived at solely on the basis of the complexity of w*, or alternatively, if we distribute across the unions, as n times the complexity of w* (where n is the number of prefix words). Obviously this latter case introduces a sort of spurious degree of complexity. The number of different (non-collinear) w* words is of course the more fundamental measure. We therefore illustrate how a w* (of degree 1) having several prefixes, can be realized by a 1-CTM having slightly generalized acceptance characteristics.

Example: 1-CRE = $(\Lambda \cup aab \cup aaaabbb)\cdot(aaabb)^*$

The starred word w (in this case (aaabb)) is multiplied by an integer p large enough so that the number of a's in p•w and the number of b's in p•w exceeds the largest number of a's and the largest number of b's of any prefix word. The cyclic counter read-head of the tape machine is then constructed on p•w rather than on w. The 1-CTM will have a final state pair for each of the prefix words.

In this example p = 2 is sufficient to enlarge w to include any of the prefix words. Thus 2(aaabb) = (aaaaaabbbb) can be employed to form counting cycle bases sufficiently large to implement the desired 1-CTM. There must be a final state pair for (aaaaaabbbb) and also a final state pair for (aaabb) (for p = 1), for (aab) and for (aaaabbb).



Acceptance: CT:0 and $[(a_0,b_0)$ or (a_2,b_1) or (a_3,b_2) or $(a_4,b_3)]$

REFERENCES

1. [A] Aanderaa, S., "On the Algebra of Regular Expressions," Appl. Math. Harvard U., Cambridge, 1965, 18 pages (ditto).

2. [B] Buchi, J. R., "Algebraic Theory of Feedback," Automata Theory, ec. E. R. Caianiello, Academic Press, 1966, pp. 70-101.

3. [C-E-W] Copi, I. M., Elgot, C. C., and Wright, J. B., "Realization of Events by Logical Nets," JACM 5, 1958, pp. 181-186.

4. [E] Elgot, C. C. et al, "Remarks on Behavior of Commutative Machines," in Mathematical Automata Theory Vol. 1 — Finite Automata (Technical Report RADC-TR-65-1133, Vol. 1).

5. [G] Give'on, Y., "Normal Monoids and Factor Monoids of Commutative Monoids," Technical Report, The University of Michigan, Ann Arbor, May 1965.

6. [G-S] Ginsburg, S., and Spanier, E., "Bounded Regular Sets," System Development Corporation Report, January, 1966.

7. [K] Kleene, S. C., "Representation of Events in Nerve Nets and Finite Automata," in Automata Studies, ed. Shannon and McCarthy, 1956, pp. 285-306.

8. [L] Laing, R., "Tape Machine Realizations of Commutative-Regular Events," Technical Report, The University of Michigan, September, 1965.

9. [L-W] Laing, R., and Wright, J. B., "Commutative Machines," Technical Report, The University of Michigan, Ann Arbor, December, 1962.

10. [M] Minsky, M., "Recursive Unsolvability of Post's Problem of "Tag" and Other Topics in Theory of Turing Machines," Annals of Math 74, 3, 1961, pp. 437-455.

11. [M-Y] McNaughton, R., and Yamada, H., "Regular Expressions and State Graphs for Automata," Transactions IRE PGEC, EC-9, No. 1, 1960, pp. 39-47.

12. [P] Perrot, J. F., "On Abelian Regular Events," Paper presented at Conference on the Algebraic Theory of Machines, Languages and Semigroups," Asilomar, Pacific Grove, Summer 1966.

13. [R] Redko, V. N., "On Defining Relations for the Algebra of Regular Events," Ukrain Mat. 2. 16, 1964, pp. 120-126.

14. [S] Salomaa, A., "Two Complete Axiom Systems for the Algebra of Regular Events," JACM 13, 1, 1966, pp. 158-169.

DISTRIBUTION LIST

(One copy unless otherwise noted)

Technical Library
Director Defense Res. & Eng.
Room 3C-128, The Pentagon
Washington, D.C. 20301

Defense Documentation Center 20
Cameron Station
Alexandria, Virginia 22314

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

Director, Naval Research Laboratory 6
Technical Information Officer
Washington 25, D.C.
Attention: Code 2000

Commanding Officer 10
Office of Naval Research
Navy 100, Fleet Post Office Box 39
New York, New York 09599

Commanding Officer
ONR Branch Office
207 West 24th Street
New York 11, New York

Office of Naval Research Branch
Office
495 Summer Street
Boston, Massachusetts 02110

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

David Taylor Model Basin
Washington,D.C. 20007
Attn: Code 042, Technical Library

Naval Electronics Laboratory
San Diego 52, California
Attn: Technical Library

Dr. Daniel Alpert, Director
Coordinated Science Laboratory
University of Illinois
Urbana, Illinois

Air Force Cambridge Research Labs
Laurence C. Hanscom Field
Bedford, Massachusetts
Attn: Research Library, CRMXL R

U. S. Naval Weapons Laboratory
Dahlgren, Virginia 22448
Attn: G. H. Gleissner, Code K4
Asst. Dir. for Computation

National Bureau of Standards
Data Processing Systems Division
Room 239, Building 10
Washington 25, D.C.
Attn: A. K. Smilow

George C. Francis
Computing Laboratory, BRL
Aberdeen Proving Ground, Maryland

Office of Naval Research
Branch Office, Chicago
230 North Michigan Avenue
Chicago, Illinois 60601

Commanding Officer
ONR Branch Office
1030 E. Green Street
Pasadena, California

Commanding Officer
ONR Branch Office
1076 Mission Street
San Francisco, California 94103

DISTRIBUTION LIST (Concluded)

The University of Michigan
Department of Philosophy
Attn: Professor A. W. Burks

National Physical Laboratory
Teddington, Middlesex, England
Attn: Dr. A. M. Uttley, Supt.
Autonomics Division

Commanding Officer
Harry Diamond Laboratories
Washington, D.C. 20438
Attn: Library

Commanding Officer and Director
U. S. Naval Training Device Center
Port Washington
Long Island, New York
Attn: Technical Library

Department of the Army
Office of the Chief of Research
and Development
Pentagon, Room 3D442
Washington 25, D.C.
Attn: Mr. L. H. Geiger

National Security Agency
Fort George G. Meade, Maryland
Attn: Librarian, C-332

Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts
Attn: Library

Office of Naval Research
Washington 25, D.C.
Attn: Code 432

Dr. Kenneth Krohn
Krohn Rhodes Research Institute, Inc.
328 Pennsylvania Avenue, S. E.
Washington 13, D. C.

Dr. Larry Fogel
Decision Science, Inc.
6508 Pacific Highway
San Diego, California

National Bureau of Standards
Applications Engineering Section
Washington 25, D. C.
Attn: Miss Mary E. Stevens

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Logic of Computers Group The University of Michigan Ann Arbor, Michigan 48104	Unclassified
	2b. GROUP

3. REPORT TITLE

REALIZATION AND COMPLEXITY OF COMMUTATIVE EVENTS

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Technical Report

5. AUTHOR(S) (Last name, first name, initial)

Laing, Richard

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
March 1967	39	14

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
Nonr 1224(21)	
b. PROJECT NO.	03105-48-T
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. AVAILABILITY/LIMITATION NOTICES

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Office of Naval Research Department of the Navy Washington, D.C.

13. ABSTRACT

A measure of complexity is defined for a class of expressions denoting behaviors of commutative machines. For an expression of complexity m there is an m tape machine which realizes in real-time the event denoted. The machine realizations can be of many kinds; the "simplest" realizations are those composed of finite counters of a single kind of input letter, combined with infinite state machines which check off numbers (mod p) of one kind of input letter against numbers (mod r) of a second kind of input letter. The state diagrams of the finite state machines are in the form of cycles (possibly with linear "lead-ins"); the state diagrams of the infinite state machines graphically are infinite cylinders (possibly with two-dimensional lead in arrays). For any machine with m counter tapes (as here defined), an m-complex expression can be obtained which denotes the event which is the behavior of the machine. Finally it is shown that for all m there are m-complex expressions which require m-counter tape machines for realizing in real-time the events denoted.

DD FORM 1 JAN 64 1473

Unclassified
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
automata						
combinatorics						
commutative machines						
real-time acceptors						
infinite state counters						
regular expressions						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2*a*. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2*b*. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7*a*. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7*b*. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8*a*. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8*b*, 8*c*, & 8*d*. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9*a*. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9*b*. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.