SU-SEL-75-023

Sequential Circuit Output Probabilities from Regular Expressions

by

Kenneth P. Parker and Edward J. McCluskey

June 1975

Technical Report #93

This work was supported in part by National Science Foundation Grant GJ-40286 and was submitted to the *IEEE Transactions on Computers.*

DIGITAL SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES

STANFORD UNIVERSITY • STANFORD, CALIFORNIA

SU-SEL-75-023

SEQUENTIAL CIRCUIT OUTPUT PROBABILITIES FROM REGULAR EXPRESSIONS*

by

Kenneth P. Parker and Edward J. McCluskey

Technical Report # 93

June 1975

Digital Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California

*This work was supported in part by National Science Foundation grant GJ-40286 and was submitted to the IEEE Transactions on Comnuters.

SEQUENTIAL CIRCUIT OUTPUT PROBABILITIES FROM REGULAR EXPRESSIONS

by

Kenneth P. Parker and Edward J. McCluskey

June 1975

ABSTRACT

This paper presents a number of methods for finding sequential circuit output probabilities using regular expressions. Various classes of regular expressions, based on their form, are defined and it is shown how to easily find multistep transition probabilites directly from the regular expressions. A new procedure for finding steady state probabilities is given which proceeds either from a regular expression or a state diagram description. This procedure is based on the concept of synchronization of the related machine, and is useful for those problems where synchronization sequences exist. In the cases where these techniques can be utilized, substantial savings in computation can be realized. Further, application to other areas such as multinomial Markov processes is immediate.

I. INTRODUCTION

Kleene, in 1956, showed that finite automata could be represented with a regular expression notation [Kleene, 1956]. McNaughton and Yamada gave algorithms showing the connection between regular expressions and state graphs, and introduced the concepts of complementation and intersection [McNaughton and Yamada, 1960]. An extensive survey of regular expressions is given by Brzozowski [Brzozowski, 1962]. In 1963, Brzozowski and McCluskey explored the use of signal flow graph techniques to obtain regular expressions from state diagrams [Brzozowski and McCluskey, 1963]. The derivation of a regular expression directly from a sequential circuit diagram is explored by Brzozowski [Brzozowski, 1964a], who also provides us with the concept of the derivative of a regular expression [Brzozowski, 1964b] and its use in forming the state diagram of the accepting automaton.

Regular expressions are a precise, unambiguous language for describing finite automata which sometimes enjoy the advantages of compactness and similarity to the word description of the accepting automaton, though too, they can be arbitrarily complex. A disadvantage is that it is often difficult to determine whether two expressions are equivalent, i.e., describe the same automaton, without reverting to state table descriptions and determining whether they are equivalent [Hennie, 1968]. This is due to the lack of a canonical form for regular expressions. However, for some problems, regular expressions are quite useful. They have been used in formal language theory as well as automata theory, and in this paper, will be used to approach some probability problems.

Probabilistic analysis has long been in use in reliability studies. Recently, interest has arisen in logic circuit analysis using probability [Parker and McCluskey, 1975b] and its application to reliability [Ogus, 1975], fault analysis in combinational circuits [Parker and McCluskey, 1975a; Shedletsky and McCluskey, 1975b; Shedletsky, 1975], and fault analysis in sequential networks [Shedletsky and McCluskey, 1975a]. The work by Shedletsky and McCluskey approaches the difficult problem of sequential circuit analysis and provides some specific methods of analysis utilizing Markov chains. This paper will also deal with sequential circuit analysis on a general level.

We are interested in the solution to a number of similar problems;

(1) the probability that a finite sequential machine, given random input, produces some specified output,

(2) the probability that a suitable[1] Markov chain reaches a certain state,

(3) the steady state probabilities of a suitable Markov process.

All of these problems can be handled using Markov chain analysis [Shedletsky and McCluskey, 1975a], but this involves solving systems of equations, inverting matrices, etc. We show that in some cases we can take advantage of regular expressions to obtain these results directly.

II. BACKGROUND

Regular expressions are defined recursively as follows: given an alphabet $A_k = \{a_0, a_1, a_2, \ldots, a_{k-1}\}$, the symbols λ and ϕ, and the regular

[1] Specifically, a Markov chain representing a multinomial process [Kemeny and Snell, 1960].

operators "+", "·", and "*" and parentheses,

(1) A string consisting of a single alphabet symbol, single λ, or single ϕ is a regular expression.

(2) If P and Q are regular expressions, then so are $(P+Q)$, $(P \cdot Q)$, and P^*.

(3) No other string of symbols is a regular expression unless it follows from (1) and (2) in a finite number of steps.

Regular expressions represent sets of sequences of symbols. The symbol λ represents the sequence of zero length. The symbol ϕ represents the null set or empty set of sequences. The operation "+" in P+Q denotes the union of the sets P and Q. . The operation "·" in P·Q (from now on written as PQ with "·" understood) is the concatenation of elements of P and Q. In general, concatenation is not commutative, i.e., PQ may not equal QP. The star operator "*" or iterate of a set is defined as the infinite union

$$P^* = A + P + PP + PPP + \ldots \tag{2.1}$$

which we abbreviate as

$$P^* = \bigcup_{i=0}^{\infty} P^i \tag{2.2}$$

where P^i is i copies of P concatenated and P^0 is defined as λ. Given that P, Q, and R are regular expressions, then the following relations exist.

$$P + Q = Q + P \qquad (+ \text{ commutative}) \qquad (2.3)$$

$$(P + Q) + R = P + (Q + R) \qquad (+ \text{ associative}) \qquad (2.4)$$

$$(PQ)R = P(QR) \qquad (\cdot \text{ associative}) \qquad (2.5)$$

$$PQ + PR = P(Q + R) \qquad (\text{left distributivity}) \qquad (2.6)$$

$$PR + QR = (P + Q)R \qquad (\text{right distributivity}) \qquad (2.7)$$

$$R + \phi = \phi + R = R \qquad (+ \text{ identity}) \qquad (2.8)$$

$$R\phi = \phi R = \phi \qquad (\cdot \text{ zero}) \qquad (2.9)$$

$$R\lambda = \lambda R = R \qquad (\cdot \text{ identity}) \qquad (2.10)$$

$$R + R = R \qquad (\text{idempotency}) \qquad (2.11)$$

$$\lambda^* = \lambda \qquad (2.12)$$

$$\phi^* = \lambda \qquad (2.13)$$

Equation 2.11 illustrates the equivalence problem. As an example [Brzozowski, 1962], consider the following expressions.

$$A = (0 + 10^*1)^*10^*$$
$$B = 0^*1(0 + 10^*1)^* \qquad (2.14)$$

At a glance they do not appear equivalent so that their union would be A + B. However, they are equivalent (they were derived by different procedures from the same machine description) so that indeed, A + B = A = B. Since later we will derive probability expressions from these regular expressions, we need to avoid counting probability masses more than once. We will require all regular expressions to be simplified via equation 2.11. In general it may not be obvious that this condition is satisfied. However, we will soon define a series of forms that regualr expressions may have which we find both useful and fairly easy to check for this simplification.

Definition 2.1 The derivative [Brzozowski, 1964b] of a regular expression R with respect to a sequence a, denoted $D_a R$, is another regular expression given by

$$D_a R = \{s \mid as \in R\} . \tag{2.15}$$

Also,

$$D_{ab} R = D_b(D_a R) \tag{2.16}$$

Some simple examples are, for R = (011 + 00)

$$D_\lambda R = R \qquad D_{011} R = \lambda$$
$$D_0 R = 11 + 0 \qquad D_1 R = \phi$$
$$D_{01} R = 1$$

Full details are given in [Brzozowski, 1964b]. Derivatives will be useful later in defining and determining substring relations.

Definition 2.2 The symbol I is reserved for the union of the alphabet symbols and constitutes a 'dont care' symbol of length 1. Then I* is the set of all possible strings.

Definition 2.3 A sequence s is defined to be a **substring** of **regular** expression R, denoted $s \in SUB(R)$, iff there exists an $x \in I^*$ such that $D_{xs} R \neq \phi$. If $s \in SUB(R)$ then for some $r \in R$, the sequence s is embedded in r. For example, 011 is a substring of 110110, since there exists an $x \in I^*$ (x = 11) such that $D_{11011}(110110) = 0 \neq \phi$. However, 111 is not a substring of 110110.

We now need to clarify the notion of an event, since it occurs both in the regular expression literature and in probability theory. We consider an event to be some application of zero or more symbols to an automaton. If the automaton, at the end of the event, produces a recognition output, i.e., it recognizes the input as a memeber of a set of inputs for which it was designed to accept, we say the automaton accepts the event. The set of events that an automaton accepts can be described by a regular expression [Kleene, 1956].

Definiton 2.4 A simple expression (or elementary expression) is the concatenation of zero or more alphabet symbols. The length of a simple expression T, denoted L(T), is an integer count of the number of symbols in T. Note L(T) = 0 implies T = λ. For example, T = abcba is a simple expression of length 5.

Definiton 2.5 A finite composite expression is the union of a finite number of simple expressions. Example: T = ab + bc + acb is a finite composite expression.

Definition 2.6 An infinite composite expression is the star iterate of a finite composite expression. Example: T = $(a + ab + abca)^*$ is an infinite composite.

Definition 2.7 A concatenate composite expression is the concatenation of two or more finite or infinite composite expressions. For example, let A,B be finite composite, C be infinite composite. Then ABC, A*B, AB^*, and A^*B^* are concatenate composite, as is AB, but AB by distributivity reduces to another finite composite.

Definition 2.8 All other expressions that are not simple, finite/infinite/ concatenate composite in form are called complex expressions. For example, $(ab^*c + a^*b)^*$ is complex. Note that $(a^*b^*)^*$ and $(a^* + b^*)^*$ are complex expressions, but by the identity

$$(a^*b^*)^* = (a^* + b^*)^* = (a + b)^* \tag{2.17}$$

they can be written as infinite composite expressions.

Definition 2.9 A delay expression is a special case of concatenate composite expression of the form $T = I^*S$ for some finite composite S.

Delay expressions are identical to 'non-initial definite events' in [Brzozowski, 1962]. The name "delay" arises from the fact that any delay expression corresponds to a delay realized sequential machine [Hennie, 1968] (see finure 4.1). The following definition differs somewhat from Brzozowski's definition of an initial event.

Definition 2.10 An initial expression is one where the beginning of the sequence of input symbols must be examined in order to determine the acceptability of the expression. For example, ab, ab*c, acb^*I^*, $(c + ab^*a)^*$ are initial expressions. Delay expressions are not initial? since the input symbols contained in I^* do not affect the acceptability of an expression.

Definition 2.11 A synchronized expression is of the form I^*SR where S is finite composite and R is a regular expression. Delay expressions are a special case. The name "synchronized" stems from the machine realization, where S represents a set of synchronizing sequences [Hennie, 1968]. To determine the acceptability of a synchronized expression, all

symbols back to the last occurrence of s ε S must be examined. Synchronized expressions are not initial.

Definition 2.12 Given an alphabet $I = a_0, a_1, \ldots, a_k$ 1, define the probability of the symbols $P(a_i)$ as a set of real numbers such that $0 \leq P(a_i) \leq 1$ for all i and $P(a_0) + P(a_1) + \ldots P(a_{k-1}) = 1$. Further, define $P(\lambda) = 1$ and $P(\phi) = 0$.

Lemma 2.1 The probability of a simple expression of length $m \geq 0$ is given by the product of the probabilities of its symbols. For example, for E = abc, the probability of this expression is P(a)P(b)P(c). We are assuming that all appearances of any symbol at any place in an event are independent of any other symbol.

This probability of an expression of length m is identical to the probability that the corresponding automaton accepts a string of length m given the probabilities (and independence assumption) of its input symbols. Most expressions that we have defined do not have a single length. For example, the finite composite E = (ab + abcab) represents a machine that can accept after 2 input symbols and after 5 input symbols. More complicated expressions become more difficult to "eyeball". The following artiface provides a convenient solution for some more complicated expressions.

Definition 2.13 Given a regular expression R, the generating function of R, GEN(R), is constructed as follows:

(1) For each simple expression within R, convert it to a probability expression via lemma 2.1 and then follow it by dummy variable t^m where m is its length.

(2) Replace each starred term E* by

$$\sum_{j=0}^{\infty} (E)^j$$

repeating (this can be done in any order), until all the stars are gone.

Example 2.1 Let R = $(ab + abca)^*$. Then GEN(R) is formed as follows.

(A) Let x=P(a), y=P(b), z=P(c), The step (1) yields

$$(xyt^2 + x^2yzt^4)^*$$

(B) Step (2) yields

$$GEN(R) = \sum_{j=0}^{\infty} (xyt^2 + x^2yzt^4)^j.$$

Example 2.2 Let R = $(a + ab^* + b^*c)^*$. Let a, b, and c be probabilities.

(A) Performing step (1) we have

$$(at + at(bt)^* + (bt)^*ct)^*.$$

(B) Removing the 3 stars gives

$$GEN(R) = \sum_{i=0}^{\infty} \left(at + at \sum_{j=0}^{\infty} (bt)^j + \sum_{k=0}^{\infty} (bt)^k ct \right)^i$$

which can be simplified due to the commutativity[2] of the variables as

$$GEN(R) = \sum_{i=0}^{\infty} \left(at + (a+c) \sum_{j=1}^{\infty} (bt)^j \right)^i$$

which is somewhat less complicated.

The next example illustrates a problem in the procedure.

Example 2.3 Assume A and B are both simple expressions. Let $R = (A + B)(A + B)$. Then $GEN(R) = (GEN(A) + GEN(B))^2 = GEN(A)^2 + 2GEN(A)GEN(B) + GEN(B)^2$. Let A = a and B = aa. Then the preceeding expression is incorrect because (a + aa)(a + aa) = (aa + aaa + aaaa) by equation 2.11. The correct result for this case is $GEN(R) = GEN(A)^2 + GEN(A)GEN(B) + GEN(B)^2$ but this cannot be determined without prior examination of A and B.

Lemma 2.2 Given regular expression R and GEN(R), the probability of accepting a string of length m is contained in the coefficient of t^m. This is found by an Algebraic derivative operation

$$\frac{1}{m!} \frac{d^m}{dt^m} GEN(R) \tag{2.18}$$

Proof Equation 2.18 at t = 0 picks out of GEN(R) the coefficient of t^m and deletes all the others. This coefficient is the desired probability. To prove this, we start with simple events P and Q with lengths j and k. It is immediate for GEN(P) and GEN(Q) that the assertion is true. Next we see that GEN(PQ) = GEN(P)GEN(Q) since the multiplication

[2] We note here that our generating function provides an inventory of probabilities, indexed by sequence length. A similar ariface can be used to inventory the sequences, by not converting simply expressions into probabilities and observing the non-commutativity of the symbols under concatenation.

of t^j and t^k produces exponent j+k which is the length of PQ, and the corresponding probabilities have been properly multiplied in either case. After allowing the assumption that $P \neq Q$ (so that we don't have the idempotency problem of equation 2.11) we see that GEN(P+Q) = GEN(P) + GEN(Q) in the procedure of definiton 2.13. Finally, we show that the star iterate is properly handled. Examination of equation 2.1, plus the preceding arguments, show that step 2 of definition 2.13 yields the proper sequence of terms for the iterate case.

For example, if $GEN(R) = \sum_{i=0}^{\infty} (ab^2t^3)^i$, the probability of accepting a string of length m = 6 is

$$\frac{1}{6!}\frac{d^6}{dt^6}\sum_{i=0}^{\infty}(ab^2t^3)^i\Big|_{t=0} = \frac{1}{6!}\sum_{i=0}^{\infty}(a^ib^{2i}\frac{d^6}{dt^6}t^{3i})\Big|_{t=0}$$

$$= a^2b^4 \quad .$$

Lemma 2.2 is a general result. However, taking higher order derivatives can quickly grow tedious. Also, we will be interested in steady state solution, i.e., where m approaches infinity. The next section provides examples of the use of lemma 2.2 and deals with specific simplifications we can enjoy based on the structure of the event in question.

III. EVENT PROBABILITIES

The following theorems give the probabilities of an automaton accepting a finite sequence of symbols with given probability assignments. The accepting probability will be denoted $P_m(E)$, for an automaton represented by expression E and sequence length m.

Theorem 3.1 If E is simple, q = L(E) and e = P(E) then for $m \geq 0$,

$$P_m(E) = \begin{cases} e \text{ if } q=m \\ 0 \text{ otherwise} \end{cases}$$

Proof The proof follows directly from lemma 2.1. ■

Theorem 3.2 If finite composite $E = (E_0 + E_1 + \ldots + E_k)$ with all E_i simple and disjoint, $q_i = L(E_i)$, and $e_i = P(E_i)$, then for $m \geq 0$,

$$P_m(E) = \sum_{i=0}^{k} \begin{cases} e_i \text{ if } q_i=m \\ 0 \text{ otherwise} \end{cases}$$

Proof If all q_i are unique, the proof follows from theorem 3.1. If two or more q_i are equal, then because the corresponding E_i are disjoint, we can add their probabilites. A rigorous proof uses generating functions of E and equation 2.18, but the above argument should be sufficient. For example, let E = (01 + 10 + 111), with P(1) = p and P(0) = q. Then $P_1(E) = 0$, $P_2(E) = 2pq$, $P_3(E) = p^3$, and $P_k(E) = 0$ for $k > 3$. ■

Theorem 3.3 If infinite composite $E = (E_0)^*$ with simple E_0, $q = L(E_0)$, and $e = P(E_0)$, then for $m \geq 0$.

$$P_m(E) = \begin{cases} e^{m/q} \text{ if } q \text{ divides } m \\ 0 \quad \text{otherwise} \end{cases}$$

Proof We have $P_m(E) = \frac{1}{m!} \frac{d^m}{dt^m} GEN(E) \Big|_{t=0} = \frac{1}{m!} \frac{d^m}{dt^m} \sum_{j=0}^{\infty} (et^q)^j \Big|_{t=0}$

$$= \sum_{j=0}^{\infty} e^j \frac{1}{m!} \frac{d^m}{dt^m} t^{qj} \Big|_{t=0} = \begin{cases} e^j \text{ if } qj = m \\ 0 \text{ otherwise} \end{cases}$$ ●

Corollary to theorem 3.3 If infinite composite $E = (E_0 + E_1 + \ldots + E_k)^*$ with all E_i simple and disjoint, all $q_i = L(E_i) = q$ and $e_i = P(E_i)$, then for $m \geq 0$,

$$P_m(E) = \begin{cases} (e_0 + e_1 + \ldots + e_k)^{m/q} & \text{if } q \text{ divides } m \\ 0 & \text{otherwise} \end{cases}$$

As an example, with $E = (aba + bba)$, letting a and b be probabilities, it follows that $P_2(E) = 0$, $P_3(E) = ba^2 + ab^2$, $P_4(E) = P_5(E) = 0$, $P_6(E) = (ba^2 + ab^2)^2$, etc.

Theorem 3.4 If infinite composite $E = (E_0 + E_1 + \ldots + E_k)^*$ with all E_i simple and unique, $q_i = L(E_i)$, $e_i = P(E_i)$, define integers $n, j_1, j_2, \ldots j_k$ and a constraint set $C_m(E)$ on these integers given as

$$C_m(E) = \left\{ n, j_1, j_2, \ldots j_k \;\middle|\; n \geq 0,\ j_i \geq 0\ \forall\ i,\ \sum_{i=1}^{k} j_i \leq n, \right.$$

$$\left. \text{and } m = nq_0 + \sum_{i=1}^{k} j_i(q_i - q_0) \right\}$$

then for $m \geq 0$,

$$P_m(E) = \sum_{n, j_1, j_2, \ldots j_k}^{C_m(E)} \binom{n}{j_1\ j_2 \cdots\ j_k} e_0^{(n - j_1 - j_2 - \ldots - j_k)}\ e_1^{j_1} e_2^{j_2} \ldots e_k^{i_k}$$

Proof We have $P_{m,E} = \frac{1}{m!} \frac{d^m}{dt^m} \text{GEN}(E) \Big|_{t=0}$

$$= \frac{1}{m!} \frac{d^m}{dt^m} \sum_{n=0}^{\infty} (e_0 t^{q_0} + e_1 t^{q_1} + \ldots + e_k t^{q_k})^n \Big|_{t=0}$$

The multinomial expansion for the power of a sum yields

$$P_m(E) = \frac{1}{m!} \frac{d^m}{dt^m} \sum_{n=0}^{\infty} \sum_{j_1, j_2, \ldots j_k}^{n} \binom{n}{j_1 j_2 \cdots j_k}$$

$$(e_0 t^{q_0})^{(n-j_1-j_2-\ldots-j_k)} (e_1 t^{q_1})^{j_1} (e_2 t^{q_2})^{j_2} \ldots (e_k t^{q_k})^{j_k} \Big|_{t=0}$$

Collecting powers of t yields

$$P_m(E) = \sum_{n=0}^{\infty} \sum_{j_1, j_2, \ldots j_k}^{n} \binom{n}{j_1 j_2 \cdots j_k} e_0^{(n-j_1-j_2-\ldots-j_k)} e_1^{j_1} e_2^{j_2} \ldots e_k^{j_k}$$

$$\frac{1}{m!} \frac{d^m}{dt^m} t^{((n-j_1-j_2-\ldots-j_k)q_0 + j_1 q_1 + j_2 q_2 + \ldots + j_k q_k)} \Big|_{t=0}$$

However,

$$\frac{1}{m!} \frac{d^m}{dt^m} t^{((n-j_1-j_2-\ldots-j_k)q_0 \sum_{i=1}^{k} j_i q_i} \Big|_{t=0}$$

$$= \begin{cases} 1 & \text{if } m = nq_0 + \sum_{i=0}^{k} j_i (q_i - q_0) \\ 0 & \text{otherwise} \end{cases}$$

and the multinomial expansion we used adds the constraints that $n \geq 0$, $j_i \geq 0$ for all i, and $n \geq j_i + j_2 + \ldots + j_k$. All elements of $C_m(E)$ satisfy exactly these constraints so the multiple summation can be replaced by a single summation over the elements of $C_m(E)$, yielding the postulated result. ■

Inspection reveals that theorem 3.3 and its corollary are a special case of theorem 3.4 if one handles the nonexistance of a set of j_i properly.

Theorem 3.4, though perhaps formidable in appearance, has some interesting properties that simplify its interpretation and application. Indeed, probabilities can sometimes be written down by inspection.

Example 3.1 Consider infinite composite $E = (ab + acbca)^*$. Since it is composed of two simple events ab and acbca with lengths 2 and 5 (and probabilities we define as ab and a^2bc^2), the constraint set $C_m(E)$ is

$$C_m(E) = \{n,j \mid n \geq 0,\ j \geq 0,\ j \leq n,\ 2n + 3j = m\}$$

and the probability of accepting a sequence of length m is

$$P_m(E) = \sum_{n,j}^{C_m(E)} \binom{n}{j} (ab)^{n-j} (a^2bc^2)^j$$

$$P_m(E) = \sum_{n,j}^{C_m(E)} \binom{n}{j} a^{n+j} b^n c^{2j} .$$

The constraint set gives for each m, a set of n and j shown in figure 3.1.

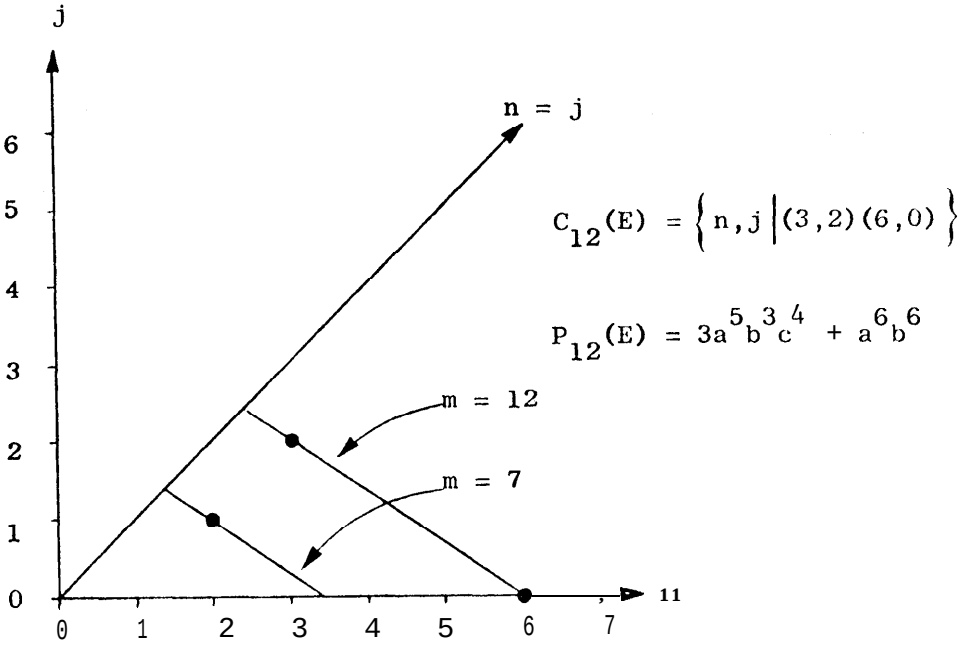


Figure 3.1

Set Cm(E) is a family of lines with slope -2/3. The integral solutions lying on or between j=0 and j=n on a given line are used in the summation of $P_m(E)$. If we think of lines as two dimensional surfaces, the constraint set gives us a family of such surfaces containing the desired (n,j). For more complicated expressions with k simple components of different lengths, we are looking for integral solutions on analagous k-dimensional surfaces.

If we examine the state diagram in figure 3.2 for the machine used in example 3.1, we notice that it is not strongly connected.

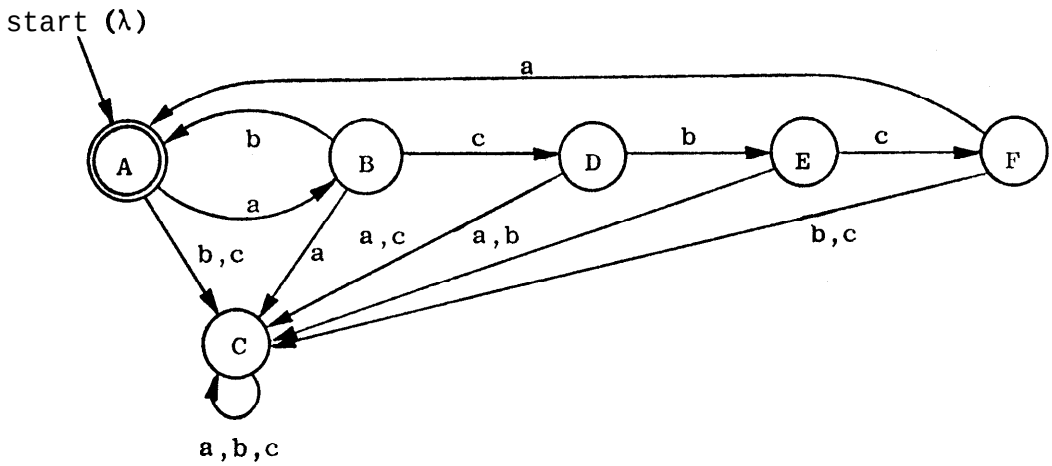


Figure 3.2

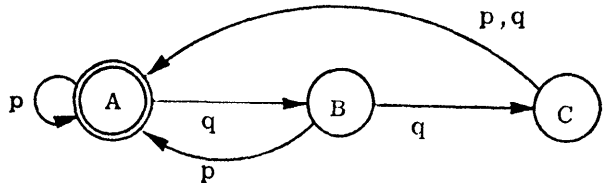
By changing our context, letting a, b, and c be probabilities summing to one, figure 3.2 represents a Markov chain (specifically, an absorbing trinomial Markov process [Kemeny and Snell, 1960]. The probability of accepting (arriving in state A) must, as time grows large, become small, or correspondingly, the likelihood that the machine is driven into the trap state C grows larger with time. Most infinite composite events will exhibit this behavior, i.e., $P_m(E) \longrightarrow 0$ for large m. This convergence

need not be monotonic with m as table 3.1 shows, for a = 0.5, b = 0.4, and c = 0.1.

Table 3.1

m	$P_m(E)$
0	1.0
1	0.0
2	0.2
3	0.0
4	0.04
5	0.001
6	0.008
7	0.0004
8	0.0016
9	0.00012
10	0.000321

Some infinite composite events correspond to strongly connected machines. For example, let I = (p + q) and infinite composite $E = (p + qp + qqp + qqq)^*$. The state diagram and state table for this event is shown in figure 3.3.



	P	q
A	A	B
B	A	C
C	A_C	A_1

$$P_m(E) = \sum_{n,j_1,j_2}^{C_m(E)} \binom{n}{j_1\ j_2} p^{n-j_2} q^{j_1+2j_2}$$

$$C_m(E) = \{n,j_1,j_2 \mid n,j_1,j_2 \geq 0,\ j_1 + j_2 \leq n,\ n+j_1+2j_2 = m\}$$

Figure 3.3

The equation for $P_m(E)$ is evaluated in table 3.2 for several values of m, with p = 0.4 and q = 0.6.

Table 3.2

m	$P_m(E)$	m	$P_m(E)$
1	0.4	11	0.509093478
2	0.4	12	0.511270261
3	0.616	13	0.509964191
4	0.4864	14	0.509964191
5	0.4864	15	0.510434376
6.	0.533056	16	0.510152265
7	0.5050624	17	0.510152265
8	0.5050624	18	0.510253825
9	0.515140096	19	0.510192889
10	0.509093478	20	0.510192889

For large values of m, the probability of accepting an infinite composite expression always converges to some value. For absorbing (not strongly connected) machines, this value is always 0 or 1. As just observed in table 3.2, the value may be somewhere in between, dependent upon the transition probability assignments. It can be shown in this example by simple Markov steady state analysis that $P_{,}(E) = 0.510204082$. Of course, evaluating $P_m(E)$ directly via theorem 3.4 is only practical for smaller m, and for larger m the steady state value $P_{,}(E)$ is usually what is desired. In the next section we deal with steady state solutions.

IV. STEADY STATE PROBABILITIES

If we consider a (finite) machine that has been receiving input for a long time we might expect its response to eventually stabilize into some pattern. Indeed this is true as a simple Markov chain analogy reveals, for any type of input that has stationary (time invariant) statistics. We denote this steady state value for expression E as $P_{,}(E)$ and show how it can be directly derived for some forms of regular expression, thereby avoiding traditional Markov analysis. We first consider delay expressions.

Theorem 4.1 Given delay expressions $E = I^*S$ where S is simple and $_4 = L(S)$, then $P_{,}(E) = P_q(S)$, that is, $P_{,}(E)$ is equal to the probability that the last q inputs formed S.

Proof Verification is trivial when the machine that realizes E is visualized as a q-stage shift register driving a decoder such as shown in figure 4.1

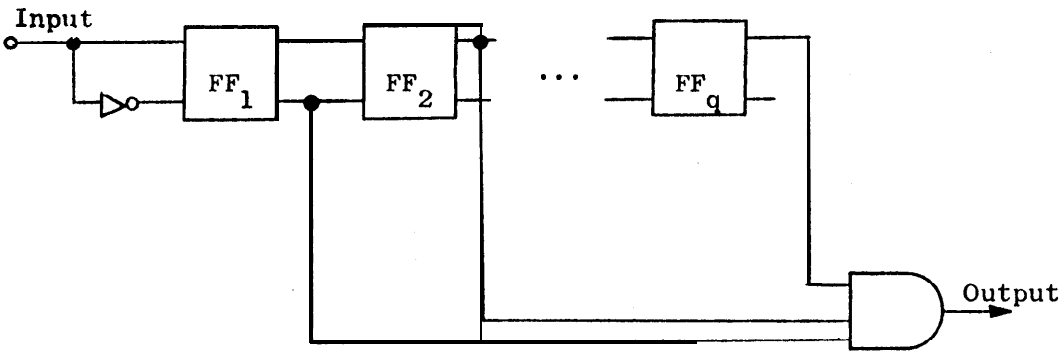


Figure 4.1

The output equals 1 if the shift register contains S. The probability of this is simply $P_q(S)$, since inputs are independent in time. ●

Corollary to theorem 4.1 Given a general delay expression $E = I^*S$ where S is finite composite $(S_1 + S_2 + \ldots + S_n)$, with simple and disjoint S_i of length $q_i = L(S_i)$, then

$$P_\infty(E) = \sum_{i=1}^{n} P_{q_1}(S_i) \quad .$$

Proof The proof is the same as for theorem 4.1 except that the output circuit (the AND gate) in figure 4.1 is replaced by a sum of products network with n AND gates, one for each S_i. Since the S_i are unique, the outputs of the n AND gates are disjoint and their probabilities simply sum for the correct result. ■

Note, care must be taken when using the corollary, that the S_i are

disjoint. For example, the following two expressions are equivalent.

$$E_1 = I^*(1 + 01 + 001) \tag{4.1}$$

$$E_2 = I^*1 \tag{4.2}$$

Since 1 is a suffix of 01 and 001 (also 01 is a suffix of 001), the S_1 in equation 4.1 are not disjoint. If s is a suffix of t then there exists an $x \varepsilon I^*$ such that xs = t. This suffix condition is a special case of substrings (definition 2.2), i.e., $D_{xs}t = \lambda$.

Now consider the expressions made up of concatenations of I, S, and T where S and T are both simple and further, $S \notin SUB(T^*)$, that is, S is not embedded in T*. Equation 4.2 lists some expressions.

$$I^*ST^* \quad (a) \tag{4.3}$$

....IIIIIS (b)

....IIIIST (c)

....IIISTT (d)

....IISTTT (e)

Quick examination show that 4.3a represents 4.2b–4.2e. Everything to the left of the rightmost occurrence of S is in essence a don't care (I) symbol in meaning. The English description of a recognizer for 4.3a is "the machine that accepts any string ending in S followed by zero or more occurrences of T". Such a machine is reminiscent of a delay machine, but for non-trivial T, cannot be so constructed with a finite number of stages. What occurred before the last appearance of S is inconsequential to the output of the machine. In figure 4.2 we show the improper[3] state

[3] In an improper state diagram, an arc may represent a transition through a number of states. In figure 4.2, the symbols S, T, U and V may be single input symbols, or more complex expressions.

diagram [Brzozowski, 1962]of a machine that recognizes I*ST*.

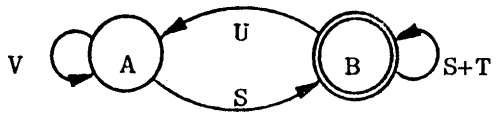


Figure 4.2

We let V be the set of all strings that do not contain an embedded S. We let U be a set of all strings that do not contain an embedded S or T. Then letting s = P(S) and t = P(T), we can find a Markov chain that gives us the probability of accepting I^*ST^*, shown in figure 4.3.

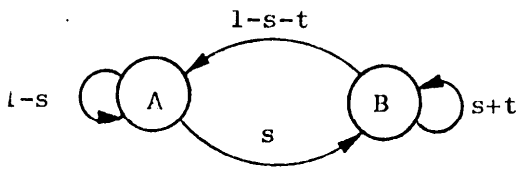


Figure 4.3

The steady state solution for state B (acceptance) for this two state chain is easily found to be

$$P(\text{state B}) = P(I^*ST^*) = s/(1-t) \qquad (4.4)$$

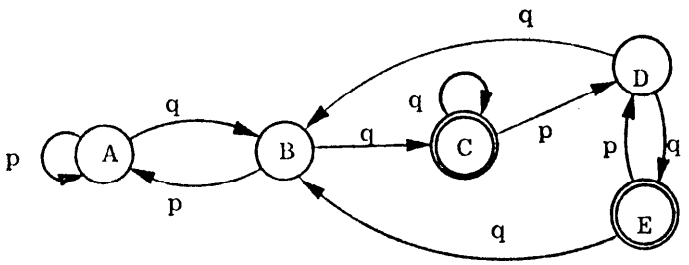
The following theorem gives this result proved by a different route, but the foregoing analysis will be useful later.

Theorem 4.2 Given simple S and T with $T \neq \lambda$, $S \notin SUB(T^*)$, $E = I^*ST^*$, $s = P(S)$ and $t = P(T)$, then $P_\infty(E) = s/(1-t)$.

Proof By theorem 4.1, the probability of receiving I^*S is s. The probability of receiving I^*ST is st; $I^*STT \rightarrow st^2$; $I^*STTT \rightarrow st^3$; etc. Since S cannot equal T, each such string is unique and has a disjoint probability of occurrence. Thus

$$P_\infty(E) = \sum_{k=0}^{\infty} st^k = s \sum_{k=0}^{\infty} t^k = s/(1-t). \quad \blacksquare$$

Example 4.1 Consider $R = I^*(00)(11)^*$. Letting q = P(0) and p = P(1), find $P_\infty(R)$. By theorem 4.2, with S = 00 and T = 11 we can immediately write $p_\infty(R)$ as $q^2/(1-p^2)$. We check this result via a Markov steady state analysis. The Markov chain for R is given in figure 4.4 along with its transition matrix.



	A	B	CD	E
A	p	q	-	-
B	p	-	q	-
C	-	-	q	p
D	-	q	-	-
E	-	q	-	P

Figure 4.4

We solve a system of equations in 4.5 derived from the matrix and the constraint that the sum of the state probabilities is unity.

$$\begin{aligned} A &= PA + qB \\ B &= qA + qD + qE \\ C &= qB + qC \\ D &= PC + pE \\ E &= pD \end{aligned} \tag{4.5}$$

Since it is easiest to solve for B we get the series of equations in 4.6.

$$\begin{aligned} A &= \frac{p}{q}B \\ B &= B \\ C &= \frac{q}{p}B \\ D &= \frac{-}{1-p^2}B \\ E &= \frac{pq}{1-p^2}B \end{aligned} \tag{4.6}$$

Their sum is

$$A + B + C + D + E = 1 = B\left(\frac{p}{q} + 1 + \frac{q}{p} + \frac{q}{1-p^2} + \frac{pq}{1-p^2}\right)$$

$$1 = B\left(\frac{p^2+pq+q^2}{pq} + \frac{q(1+p)}{1-p^2}\right)$$

$$1 = B\left(\frac{1-pq}{pq} + 1\right)$$

$$1 = B\,\frac{1}{pq}$$

So B = pq and then $C = q^2$ and $E = \frac{p^2 q^2}{1-p^2}$. Then

$$P_\infty(R) = C + E = q^2 + \frac{p^2 q^2}{1-p^2}$$

$$= q^2 \left(1 + \frac{p^2}{1-p^2} \right)$$

$$= q^2/(1-p^2)$$

as before.

Corollary to theorem 4.2 Given finite composite $S = (S_1 + S_2 + \ldots + S_k)$ with probability s and some $T \neq \lambda$ such that we can find its probability expression t. Then if for all i, $S_i \notin SUB(T^*)$ then $P(I^*ST^*) = s/(1-t)$ by the same argument used for theorem 4.2.

Example 4.2 Consider the expression $R = I^*(001 + 11)(01)^*$. A quick examination of R shows that the corollary to theorem 4.2 applies yielding

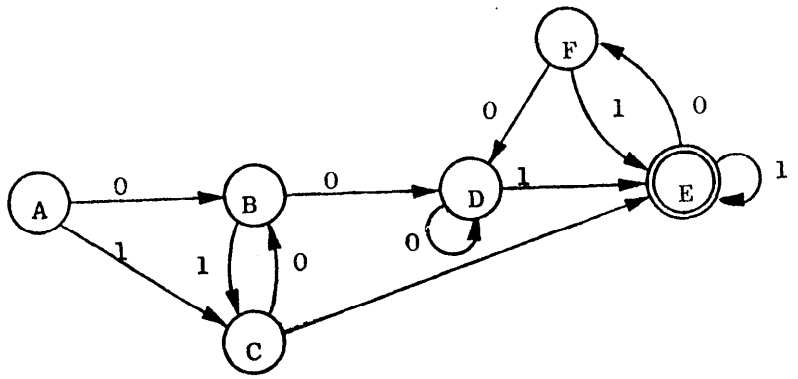
$$P_\infty(R) = (p^2 + pq^2)/(1 - pq)$$

$$= p(p + q^2)/(1 - pq)$$

$$= p(1 - pq)/(1 - pq)$$

$$= p.$$

We can derive this answer from the state description of T as shown in figure 4.5, without Markov analysis.



	0	1
A	B	C
B	D	C
C	B	E
D	D	E
E	F	E
F	D	E

Figure 4.5

From the state table we can derive the set of synchronizing sequences from the ambiguity tree [Hennie, 1968] as in figure 4.6.

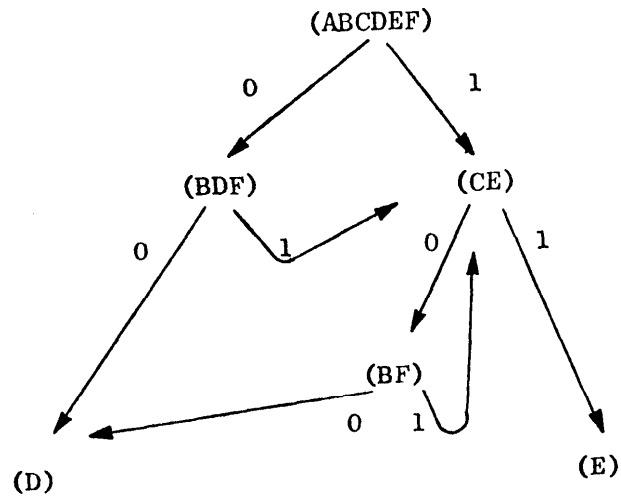


Figure 4.6

From this tree we see the set of sequences is given by

$$(00+11+011+100+1(01)^*1+1(01)^*00+01(01)^*1+01(01)^*00) \qquad (4.8)$$

but we notice that each sequence in the set ends with either 00 or 11, which are themselves synchronizing sequences. We retain only these so that the synchronizing set S is (00 + 11). Next we note that the sequence 00 leaves us in state D while sequence 11 leaves us in state E, an acceptor state. We can get from D to E with a 1, so we modify the synchronizing set so that the end result is unambiguous and has an accepting output. This gives us S = (001 + 11) which guarantees to place us in state E. Next we determine A, the set of all sequences that take us from E back to E. These are

$$A = (1+(01)+0(10)^*0(0)^*1)^* \qquad (4.9)$$

Now we examine A and determine if a synchronizing sequence from S is embedded in it. First, 11 ε SUB(A) until we remove the 1 inside. Next, 001 ε SUB(A) until we remove the term O(lO)*O(O)*l. This leaves us with $T^* = (01)^*$, which is the set of all sequences that take us from E to E without any embedded synchronizing sequences. Hence, with $s = p^2 + pq^2$ and t = pq we get $P_{\infty}(R) = (p^2 + pq^2)/(1 - pq) = p$ as before. What we have done is to construct a simple Markov chain of the form in figure 4.3 directly from the state description of the machine, for which the steady state solution is easily found. These results can be verified by a standard steady state Markov analysis of the original state diagram.

The procedure just outlined is general in that we may not necessarily be working with a regular expression that has the form of a synchronized

expression. Recall the expression in figure 3.3, $E = (p + qp + qqp + qqq)^*$. We have already seen that it converges to a steady state value other than 0 or 1. Quick inspection of its graph shows that its synchronizing set is S = (p) and that T (minus embedded synchronizing sequences) is T = (qqq). Therefore its steady state equation should be $P(E) = p/(1-q^3)$ which evaluates to 0.510204082 for p = 0.4 and q = 0.6. This agrees with the Markov solution (See section 3).

v. CONCLUSION

Utilizing regular expressions we have found new ways to calculate two probabilities; the probability of traveling from state X to state Y in exactly m steps and the steady state probabilities of being in a state. We have shown examples in finding output probabilities for sequential circuits and in multinomial Markov processes. In particular, we have shown the usefulness of synchronizing sequences in determining steady state probabilities. In some cases, the techniques developed here lead to substantial simplifications, especially in steady state problems.

Basic intuition is quite useful in approaching cases which are not directly covered in this work. For instance, if presented with a simple expression A cascaded with a synchronized expression, such as AI^*ST^*, a steady state solution for this case should quickly be seen as as/(l-t). Other problems of seeming difficulty yield when the underlying state diagrams are viewed. Usually, a flexible approach will give surprisingly good results.

Certain problems restrict the generality of these results. For example, regular expressions that have the complex form may be intractable. If the underlying automata has no synchronizing sequence, then by default, the steady state analysis cannot be undertaken. Further problems arise from the equivalence problem (equation 2.11) and determining if the substring relation holds. In many cases, this is not too difficult "by eyeball" due to the pattern recognition abilities of the eye. However, a program to perform this task could easily consume large quantities of time. Thus we find the procedures developed here to be specialized to certain tasks, for which they can be quite useful.

REFERENCES

[Brzozowski, 1962] Brzozowski, J. A., "A Survey of Regular Expressions and their Applications", IRE Trans. on Electronic Computers, Vol. EC-9, pp. 39-47, March 1960.

[Brzozowski and McCluskey, 1963] Brzozowski, J. A. and E. J. McCluskey, 'Signal Flow Graph Techniques for Sequential Circuit State Diagrams", IEEE Trans. on Computers, April 1963.

[Brzozowski, 1964a] Brzozowski, J. A., "Regular Expressions from Sequential Circuits", IEEE Trans on Computers, December 1964.

[Brzozowski, 1964b] Brzozowski, J. A., "Derivatives of Regular Expressions". Journal of ACM, Vol. 11, No. 4, October 1964.

[Hennie, 1968] Hennie, F. C., Finite State Models for Logical Machines, Wiley, 1968.

[Kemeny and Snell, 1960] Kemeny, J. and J. Snell, Finite Markov Chains, Princeton, Van Nostrand Co. 1960.

[Kleene, 1956] Kleene, S. C., "Representation of Events in Nerve Nets and Finite Automata", in Automata Studies, Annals of Mathematical Studies, Shannon and McCarthy eds., Princeton University Press, pp. 3-41, 1956.

[McNaughton and Yamada, 1960] McNaughton, R. and H. Yamada, "Regular Expressions and State Graphs for Automata", IRE Trans. on Electronic Computers, Vol. EC-9, pp. 39-47, March 1960.

[ogus, 1975] Ogus, R. C., "The Probability of a Correct Output from a Combinational Circuit", IEEE Trans. on Computers, May 1975.

[Parker and McCluskey, 1975a] Parker, K. P. and E. J. McCluskey, "Analysis of Logic Circuits with Faults Using Input Signal Probabilities", IEEE Trans. on Computers, May 1975.

[Parker and McCluskey, 1975b] Parker, K. P. and E. J. McCluskey, "Probabilistic Treatment of General Combinational Networks", IEEE Trans. on Computers, June 1975.

[Shedletsky and McCluskey, 1975a] Shedletsky, J. J. and E. J. McCluskey, "The Error Latency of a Fault in a Sequential Digital Circuit", Technical Note 56, Digital Systems Laboratory, Stanford, California, December 1974.

[Shedletsky and McCluskey, 1975b] Shedletsky, J. J. and E. J. McCluskey, "The Error Latency of a Fault in a Combinational Digital Circuit", Digest, 1975 Symposium on Fault Tolerant Computing, Paris, June 1975.

[Shedletsky, 1975] Shedletsky, J. J. , "A Rationale for the Random Testing of Digital Circuits", Digest, CompCon 75 Fall, Washington D. C., September 1975.