REPORT R-374 APRIL, 1968

COORDINATED SCIENCE LABORATORY

ON THE EQUIVALENCE OF REGULAR EXPRESSIONS

JOHN P. HAYES

UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

This work was supported in part by the Joint Services Electronics Program (U.S. Army, U.S. Navy, and U.S. Air Force) under Contract DAAB-07-67-C-0199; and NSF GK-1663.

June 5, 1968

Replacement pages for CSL Report R-374, On the Equivalence of Regular Expressions, by John P. Hayes.

Replace pages 2, 16, and 17 by the attached sheets.

An alternate method has been suggested by Brzozowski [2] which involves the direct computation of a sufficient set of derivatives of $R \oplus S$. Such computation becomes very unwieldy when even moderately long regular expressions are being considered. Also it is necessary to repeatedly compare each new derivative to all preceding ones to determine when the procedure may be halted. This approach has the advantage however that it is directly applicable to extended regular expressions.

In this paper, a relatively simple checking procedure involving only the generation and manipulation of regular equations, is described. It is applicable to extended regular expressions and is suitable for checking the equivalence of very long expressions. Due to its algebraic nature, the method is very suited to computer implementation.

A modification of this algorithm involving minimization of the derivative equations of the given regular expressions is also described. This is of interest as it leads to a canonical form for equivalent expressions.

II. OUTLINE OF THE CHECKING PROCEDURE

Suppose two regular expressions R and S are to be checked for equivalence. There are three main steps in the procedure:

1. Generation of sets of "transition equations" (called T-equations) from both R and S.

R in the usual manner. Then λ is eliminated from all equations in which it occurs and is added to all equations which did not originally contain λ. The resulting equations are the D-equations of $\sim R$.

Thus as we implicitly defined binary functions of two sets of D-equations, we can also define the negation (complement) of a set of D-equations. The significant point here is the interpretation of $\delta(X_i)$: the "negation" of λ is $\emptyset$ and the "negation" of $\emptyset$ is λ.

We now turn to the problem of checking the equivalence of extended regular expressions. These may be conveniently divided into two classes:

(A) Expressions which are some Boolean function F of restricted expressions $R_1, \ldots, R_n$. These are denoted by $F(R_i)$ or F, e.g., $10^* \oplus \sim 001$.

(B) Expressions formed from expressions of type A by means of concatenation, iteration, Boolean operations or any combination of these, e.g., $(\sim 10^*1)^*$, $((\sim 1)(1 \& 01^*))^*$.

First, the D-equations of each restricted expression R_i are separately determined; this results in n sets of D-equations. In the case of type A expressions, these equations are directly combined to give the following composite equation:

$$F(R_i) = a_1(D_{a_1}F(R_i)) + \ldots + a_k(D_{a_k}F(R_i)) + \delta(F(R_i)).$$

This gives rise to further equations just as in the case of $f(R_1,R_2)$. $\delta(D_tF(R_i))$ which is a function of λ and $\emptyset$ only, is evaluated by applying the normal rules of Boolean algebra with the exception of the special interpretation of $\sim$.

In the case of type B expressions, it is necessary to find the D-equations of expressions of the form $F_1 \cdot F_2$ and F_1^* where F_1 and F_2 are of type A. The D-equations of the F_i's are separately derived. In the case of $F_1 \cdot F_2$ we can write

$$F_1 \cdot F_2 = a_1(D_{a_1}F_1)F_2 + \dots + a_k(D_{a_k}F_1)F_2 + (\delta F_1)F_2 \tag{24}$$

which gives rise to further D-equations in the usual manner. If $\delta F_1 = \lambda$, then (24) may be non-deterministic, i.e., of the form of equation (10), and must be reduced to deterministic form as previously described.

In the case of $R = F_1^*$ we can write $R = F_1R + \lambda$ which has the same form as (24) and thus leads to a set of D-equations for F_1^*. In general type B expressions are much more cumbersome than those of type A.

<u>Example 4</u>: Suppose $R = \sim(01^* \ \& \ 0^*1)$ where & denotes intersection. Let $S = 01^*$ and $T = 0^*1$. The D-equations of S and T are

$$\begin{array}{ll} S = 0A + 1B & T = 0T + 1C \\ A = 0B + 1A + \lambda & C = 0D + 1D + \lambda \\ B = 0B + 1B & D = 0D + 1D. \end{array}$$

These equations are now combined to form the D-equations of $\sim(S \ \& \ T)$.

$$\begin{array}{l} \sim(S \ \& \ T) = 0(\sim(A \ \& \ T)) + 1(\sim(B \ \& \ C)) + \sim(\phi \ \& \ \phi) \\ \sim(A \ \& \ T) = 0(\sim(B \ \& \ T)) + 1(\sim(A \ \& \ C)) + \sim(\lambda \ \& \ \phi) \\ \sim(B \ \& \ C) = 0(\sim(B \ \& \ D)) + 1(\sim(B \ \& \ D)) + \sim(\phi \ \& \ \lambda) \\ \sim(B \ \& \ T) = 0(\sim(B \ \& \ T)) + 1(\sim(B \ \& \ C)) + \sim(\phi \ \& \ \phi) \\ \sim(A \ \& \ C) = 0(\sim(B \ \& \ D)) + 1(\sim(A \ \& \ D)) + \sim(\lambda \ \& \ \lambda) \\ \sim(B \ \& \ D) = 0(\sim(B \ \& \ D)) + 1(\sim(B \ \& \ D)) + \sim(\phi \ \& \ \phi) \\ \sim(A \ \& \ D) = 0(\sim(B \ \& \ D)) + 1(\sim(A \ \& \ D)) + \sim(\lambda \ \& \ \phi) \end{array} \tag{25}$$

ON THE EQUIVALENCE OF REGULAR EXPRESSIONS

by

John P. Hayes

Coordinated Science Laboratory
University of Illinois
Urbana, Illinois 61801

ABSTRACT

Some existing techniques for checking the equivalence of regular expressions are examined and their shortcomings discussed. A checking procedure based on the construction of regular equations is described. First, a set of transition or T-equations is obtained from the given expressions R and S. These are then converted to derivative or D-equations. Finally, a set of D-equations corresponding to $R \oplus S$ is constructed. It is shown that $R = S$ if and only if the D-equations of $R \oplus S$ do not contain the empty string λ.

Upper bounds are determined for the number of equations involved in each step of the procedure. Unlike previously-described methods, neither the construction of graphs nor the computation of derivatives is required. This technique is also applicable to extended regular expressions involving any Boolean operators.

A modified procedure involving minimization of the D-equations of R and S is also discussed. $R = S$ if and only if their respective minimized D-equations are isomorphic. Finally, it is shown that these equations lead to a canonical form for equivalent expressions.

KEY WORDS AND PHRASES: regular expressions, equivalence, regular equations, derivatives of regular expressions, extended regular expressions, transition graphs, finite-state machines, minimization, canonical forms.

CR CATEGORIES: 5.22

I. INTRODUCTION

There are several approaches to the problem of determining if two regular expressions are equivalent. In certain cases, it is possible by application of the basic laws of regular algebras (e.g., the distributive laws for union and concatenation), and certain simple identities (e.g., $RR^* + \lambda = R^*$) to convert the given expressions to forms which are identical. Another approach is to check if all the elements of one are in the other and vice versa. These techniques are only applicable when the regular expressions are either very short or have a relatively simple structure.

General methods of testing for equivalence invariably exploit the relationships between regular expressions, the derivatives of the regular expressions, regular equations and transition graphs.

Ginzburg's algorithm [1] requires construction of a transition graph from each of the given expressions R and S. A tabular method is then used to determine the characteristic derivatives of R and S from the transition graphs. Pairs of derivative equations are then constructed from which equivalence may be checked by inspection.

This procedure is unsatisfactory in several respects. The method of constructing the transition graph is not clearly defined. The use of graphs and tables as well as regular equations makes the procedure rather complicated and difficult to implement on a computer. Furthermore, it is not possible to treat extended regular expressions involving any Boolean operators.

An alternate method suggested by Brzozowski [2] requires the direct computation of the characteristic derivatives of $R \oplus S$. To employ this approach, it is necessary to be able to check the equivalence of the derivatives; this appears to lead to a vicious circle.

The methods mentioned above become extremely unwieldy when even moderately long regular expressions are being compared. In no case are bounds given on the number of steps required.

In this paper, a relatively simple checking procedure involving only the generation and manipulation of regular equations, is described. It is applicable to extended regular expressions and is suitable for checking the equivalence of very long expressions. Due to its algebraic nature, the method is very suited to computer implementation.

A modification of this algorithm involving minimization of the derivative equations of the given regular expressions is also described. This is of interest as it leads to a canonical form for equivalent expressions.

II. OUTLINE OF THE CHECKING PROCEDURE

Suppose two regular expressions R and S are to be checked for equivalence. There are three main steps in the procedure:

1. Generation of sets of "transition equations" (called T-equations) from both R and S.

2. Conversion of the T-equations to derivative equations (called D-equations).
3. Construction of the D-equations of $R \oplus S$ from the D-equations of R and S. It is shown that $R = S$ if and only if λ does not appear in the D-equations of $R \oplus S$.

The procedure can be modified by replacing Step 3 above by: reduction of the D-equations of R and S to a form isomorphic to their respective characteristic equations. It is proved in [3] that to every regular expression there corresponds a unique set of characteristic equations. From this it follows that $R = S$ if and only if the reduced D-equations are isomorphic.

III. THE TRANSITION EQUATIONS

Suppose two regular expressions R and S over the alphabet $Z = \{a_1, a_2, \ldots, a_k\}$ are to be compared. R and S are assumed to be in "restricted" form, i.e., to contain only the operators concatenation (denoted by $\cdot$ or juxtaposition), logical union (+) and iteration (*). This restriction will be lifted later.

A left-linear term is an expression of the form $a_i X_j$ where $a_i \in Z$ or $a_i = \lambda$, and X_j is a symbol denoting a non-empty regular expression. A set of left-linear equations, i.e., equations containing only the union of left-linear terms, is derived from each of the given expressions. The technique described is essentially that used by Kuck [4] to obtain state diagrams from regular expressions.

The given expression R is denoted by some symbol X_1 and the equation

$$X_1 = R$$

is written. This equation is systematically converted to left-linear form by introducing new symbols X_i to denote sub-expressions in R. Each new "unknown" X_i introduces another equation which is in turn reduced to left-linear form. A finite set of equations is eventually obtained. The procedure is best described by the following set of rules which are applied in sequence to the given regular expression:

A. Equation Rules

Consider a regular equation of the form

$$X_i = R \qquad (1)$$

A1: If R is already left-linear, (1) is left unchanged.

A2: If R is of the form aR_1 where $a \in Z$, replace (1) by

$$X_i = aX_j$$

where X_j is a new symbol, and introduce the equation

$$X_j = R_1 \qquad (2)$$

The equation rules are now applied to (2).

<u>A3</u>: If R has the form $R_1^*R_2$ then using Arden's Theorem, (1) is replaced by

$$X_i = R_1X_i + R_2 \tag{3}$$

The term rules are now applied to each term on the right-hand side of (3).

<u>A4</u>: If the foregoing rules are not applicable, R must be in the form $(R_1 + \dots + R_n)R_t$ where R_t may be implicitly λ. If R_t is some previously introduced unknown X_t, then the distributive laws are used to replace R by $R_1X_t + \dots + R_nX_t$. The term rules are then applied to each R_iX_t. If R_t is not an unknown then a new unknown X_t is introduced to denote it. R is now replaced by $R_1X_t + \dots + R_nX_t$ and the term rules are applied to each R_iX_t. Furthermore, a new equation

$$X_t = R_t \tag{4}$$

is added. The equation rules are applied to (4).

<u>B. Term Rules</u>

These rules are applied in turn to each term R obtained by application of either the equation or term rules.

<u>B1</u>: If R is left-linear, it is left unchanged.

<u>B2</u>: If R is of the form aR_1 where $a \in Z$, replace it by aX_j and introduce the equation

$$X_j = R_1$$

to which the equation rules are applied.

<u>B3</u>: If R is of the form $R_1^*R_2$, replace it by X_j and add the equation

$$X_j = R_1X_j + R_2 \tag{5}$$

The term rules must now be applied to the terms on the right-hand side of (5).

<u>B4</u>: R must have the form $(R_1 + \ldots + R_n)R_t$ where R_t may be λ. If R_t is already some unknown X_t, then R is replaced by $R_1X_t + \ldots + R_nX_t$, and the term rules are applied to each term R_iX_t. Otherwise, X_t must be introduced to denote R_t. R is replaced by $R_1X_t + \ldots + R_nX_t$, the term rules are applied to each R_iX_t and the equation

$$X_t = R_t \tag{6}$$

is introduced. The equation rules must be applied to (6).

A simple upper bound on N, the number of equations generated, may be deduced from the rules themselves and from the number of operators in the given expression R. Let N_u and N_c denote the number of occurrences in R of the union and concatenation operators respectively. Each concatenation in R introduces at most one unknown (from rules A2 and B2). Each union operator gives rise to at most two terms, each of which may require introduction of a new unknown; in addition, the unknown X_t may be introduced (rules A4 and B4). Thus at most $2N_u + 1$ unknowns can be associated with the union operators. Finally it can be seen that no unknowns are introduced by the iteration operators (rules A3 and B3). An upper bound on N is therefore given by

$$N \leqq N_c + 2\, N_u + 2 \qquad (7)$$

where an extra 1 is added to account for X_1.

In general, (7) gives a loose upper bound on the number of equations generated (it becomes an equality in certain cases, e.g., when $R = 1^* + 0^*$). A formula giving N exactly would depend on the structure of R in a very complex manner and is probably not worth deriving.

A transition graph for R may be constructed directly from these equations; each unknown X_i corresponds to a state on the graph. This method of construction is more truly "mechanical" than the well-known algorithm [5] referred to by Ginzburg [1] and it gives rise to fewer states. From their close relationship to transition graphs, the equations will be referred to as <u>transition</u> or <u>T-equations</u>.

The T-equations may also be regarded as a set of productions representing the given expression R. In this case, "=" denotes the replacement symbol "$\rightarrow$" and the X_i are non-terminal symbols in the phrase-structure grammer specifying R.

<u>Example 1</u>: Consider the regular expression $R = (01^*0)^*01^*$ over the alphabet $\{0,1\}$. Application of A3 gives

$$R = A = 01^*0A + 01^*$$

Using B2, define $B = 1^*0A$ and $C = 1^*$. Finally, we get the T-equations

$$\begin{aligned} R = A &= 0B + 0C \\ B &= 0A + 1B \\ C &= 1C + \lambda \end{aligned} \qquad (8)$$

IV. THE DERIVATIVE EQUATIONS

Every T-equation has the following general form:

$$X_s = a_1 X_a + a_1 X_b + \ldots + a_m X_n + X_p + \ldots + X_q + \delta(X_s) \tag{9}$$

where each $a_i \in Z$, X_j is an unknown and $\delta(X_s) = \lambda$ or ϕ. All terms of the form X_j on the right-hand side of (9), i.e., all unknowns not preceded by some a_i, are replaced by the right-hand side of the equation for the particular X_j. This process is repeated until all X_j terms have been eliminated. There are at most N -1 such terms. If this substitution results in repetition of terms in the same equation, then all occurrences except one of the repeated term are eliminated. This change does not affect the equality. Thus the T-equations are reduced to the following form

$$X_s = a_1 X_a + a_1 X_b + \ldots + a_m X_n + \delta(X_s) \tag{10}$$

Some unknowns may no longer occur on the right-hand side of any T-equation. The equations corresponding to such unknowns may be eliminated.

Next, all terms in the equation for X_1 (which denotes the original expression R) beginning with the same a_1 are grouped together as follows:

$$X_1 = a_1 \sum_i X_i + a_2 \sum_i X_i + \ldots + a_m \sum_i X_i + \delta(X_1)$$

Replace each group of more than one unknown by some new symbol Y_j. This gives

$$X_1 = a_1Y_1 + a_2Y_2 + \ldots + a_mY_m + \delta(X_1) \tag{11}$$

A new equation must be added for each distinct Y_j. This is formed by taking the union of all the equations corresponding to the X_i's which form Y_j. This added equation may be in the same form as (10). If so, it is reduced to the form of (11) by grouping unknowns and introducing new symbols as needed. This process continues until no new symbols are required. Clearly the number of new symbols (and equations) which may be produced is bounded by

$$\sum_{i=2}^{N} C_i^N = 2^N - (N + 1) \tag{12}$$

where N is the initial number of T-equations.

Thus the T-equations have been transformed into a set of at most $2^N - 1$ equations of the form

$$Y_s = a_iY_1 + a_2Y_2 + \ldots + a_rY_r + \delta(Y_s) \tag{13}$$

It follows from the definition of a derivative of a regular expression, that each Y_i in (13) denotes the derivative $D_{a_i}(Y_s)$. The new equations will henceforth be referred to as <u>derivative</u> or <u>D-equations</u>. If there is no term in (13) corresponding to some $a_j \in Z$, then $D_{a_j}(Y_s) = \phi$; in this case the equation is said to be "incompletely specified". The D-equations are completely specified by defining a symbol $Y_d = \phi$. The term $a_i Y_d$ is added to all D-equations not already containing an a_i-term, and the equation

$$Y_d = a_1 Y_d + a_2 Y_d + \ldots + a_k Y_d$$

is appended to the set of D-equations, bringing the maximum number possible to 2^N. (Y_d corresponds to a dead state in the finite-state machine which accepts R.)

Let U be the set of unknowns in the completely specified D-equations obtained from R. Every derivative of each X in U is some other element of U. Hence U contains all the characteristic derivatives of R. In general, not all the unknowns in U are distinct, i.e., some $X, Y \in U$ may denote equivalent derivatives of R.

The bound 2^N on the number of D-equations generated seems rather forbidding at first glance. However, a similar but even larger bound applies to the method by which the characteristic derivatives are obtained in Ginzburg's algorithm. For if a transition graph of N' states is obtained from R, Ginzburg's derivative table may have up to $2^{N'}$ distinct patterns of check marks. Furthermore, to ensure that all the characteristic derivatives have been obtained, it is necessary to examine many derivatives which have the same pattern of check marks. As mentioned already, N' is generally greater than N. Finally, it should be noted that in many examples the number of D-equations, N_d, is much less than 2^N.

The D-equations of R are closely related to the ~~state~~ diagram representation of a deterministic finite-state machine which accepts R. This state diagram is obtained from the D-equations by associating each unknown Y_i with a particular state. The unknowns whose equations contain λ correspond to accepting states.

Consider again Example 1. The first equation in (8) may be rewritten as

$$A = 0(B + C) \tag{14}$$

Letting D denote (B + C), we get from the equations for B and C:

$$\begin{aligned} D = (B + C) &= 0A + 1(B + C) + \lambda \\ &= 0A + 1D + \lambda \end{aligned}$$

Since (14) is not completely specified, we introduce the unknown E corresponding to ϕ. The final set of D-equations is

$$\begin{aligned} R = A &= 0D + 1E \\ D &= 0A + 1D + \lambda \\ E &= 0E + 1E \end{aligned} \tag{15}$$

In this example, $N_d = 3 = N$. In the following example however, N_d is quite close to the maximum possible, 2^N.

<u>Example 2</u>: Let $R = (0 + (0 + 1)0^*10^*1)^*$. The T-equations obtained by application of the equation and term rules are

$$\begin{aligned} A &= 0A + 0B + 1B + \lambda \\ B &= 0B + 1C \\ C &= 0C + 1A \end{aligned} \tag{16}$$

Thus N is again 3. The unknowns are grouped to yield the following set of D-equations:

$$
\begin{aligned}
A &= 0(A + B) + 1B + \lambda \\
B &= 0B + 1C \\
C &= 0C + 1A \\
(A + B) &= 0(A + B) + 1(B + C) + \lambda \\
(B + C) &= 0(B + C) + 1(A + C) \\
(A + C) &= 0(A + B + C) + 1(A + B) + \lambda \\
(A + B + C) &= 0(A + B + C) + 1(A + B + C) + \lambda
\end{aligned} \tag{17}
$$

Hence there are 7 D-equations, whereas $2^N = 8$. It can be shown (using the reduction technique described in Section VII) that none of the equations in (17) is redundant.

V. THE D-EQUATIONS OF $R \oplus S$

The D-equations specifying $R \oplus S$ are constructed from the D-equations of R and S. It is proved in [3] that for every Boolean operator f, $D_t(f(R,S)) = f(D_tR, D_tS)$ for all $t \in Z^*$. If f is exclusive-OR, then for all $t \in Z^*$

$$D_t(R \oplus S) = D_tR \oplus D_tS \tag{18}$$

Now $R \oplus S$ can be written in the following form:

$$R \oplus S = a_1D_{a_1}(R \oplus S) + \ldots + a_kD_{a_k}(R \oplus S) + \delta(R \oplus S)$$

Using (18), this equation may be rewritten as

$$R \oplus S = a_1(D_{a_1}R \oplus D_{a_1}S) + \ldots + a_k(D_{a_k}R \oplus D_{a_k}S) + \delta(R \oplus S) \tag{19}$$

Suppose that in the D-equations for R and S, X_{r_i} denotes $D_{a_i}R$ and

X_{s_i} denotes $D_{a_i} S$. Then (19) is equivalent to

$$R \oplus S = a_1(X_{r_1} \oplus X_{s_1}) + \ldots + a_k(X_{r_k} \oplus X_{s_k}) + \delta(R) \oplus \delta(S) \qquad (20)$$

This equation can be formed directly from the D-equation of R and S. Another equation is added for each unknown $(X_{r_i} \oplus X_{r_i})$ on the right-hand side of (20) which differs from $(R \oplus S)$; this new equation is obtained by forming the exclusive-OR combination of the equations of X_{r_i} and X_{s_i}. This process is repeated until no new unknowns are generated. The result is a set of D-equations for $R \oplus S$. There are at most $N_r N_s$ such equations, where N_r and N_s are the numbers of D-equations of R and S respectively.

The test for equivalence is based on the theorem which follows. First a simple lemma is stated without proof.

Lemma: Two regular expressions R and S over alphabet Z are equivalent if and only if $D_t R = D_t S$ for all $t \in Z^*$.

Theorem 1: Two regular expressions R and S are equivalent if and only if none of the D-equations of $R \oplus S$ contains λ.

Proof: (i) Suppose $R = S$. Then by the lemma, $D_t R = D_t S$ for all $t \in Z^*$. Thus if $\lambda \in D_t R$, then $\lambda \in D_t S$ and if $\lambda \notin D_t R$, then $\lambda \notin D_t S$. This implies that $\delta(D_t R) \oplus \delta(D_t S) = \phi$ for all $t \in Z^*$. Hence λ cannot appear in any D-equation of $R \oplus S$.

(ii) Suppose none of the D-equations of $R \oplus S$ contains λ. Since the set of unknowns includes all characteristic derivatives of $R \oplus S$, we can say that $\delta(D_t R) \oplus \delta(D_t S) = \phi$ for all $t \in Z^*$. Hence $\lambda \in D_t R$ if and only if $\lambda \in D_t S$, which means that $t \in R$ if and only if $t \in S$. It follows that $R = S$. Q.E.D.

Thus, having generated the D-equations of R and S, construct the D-equations of $R \oplus S$. If λ appears in any of these equations then $R \neq S$. If λ never appears, then we conclude that $R = S$.

Example 3: Consider the regular expression

$$S = (01^*0(011^*0)^*00)^*(01^*0(011^*0)^*(011^* + 0) + 01^*)$$

Application of the equation and term rules yields

$$\begin{aligned}
S = A &= 0B + 0C + 0D \\
B &= 0E + 1B \\
C &= 0J + 1C \\
D &= 1D + \lambda \\
E &= 0F + 0G \\
F &= 1H \\
G &= 0A \\
H &= 0E + 1H \\
J &= 0K + 0L + 0M \\
K &= 1N \\
L &= 1P \\
M &= \lambda \\
N &= 0J + 1N \\
P &= 1P + \lambda
\end{aligned} \tag{21}$$

Thus $N = 14$. However, the T-equations (21) give rise to only 6 D-equations. These are (with the unknowns renamed for brevity)

$$\begin{aligned}
S = F &= 0G + 1L \\
G &= 0H + 1G + \lambda \\
H &= 0J + 1L \\
J &= 0F + 1K + \lambda \\
K &= 0H + 1K + \lambda \\
L &= 0L + 1L
\end{aligned} \tag{22}$$

We now compare S with R in Example 1. The D-equations of $R \oplus S$ are found from (15) and (22) to be

$$
\begin{aligned}
(R \oplus S) = (A \oplus F) &= 0(D \oplus G) + 1(E \oplus L) \\
(D \oplus G) &= 0(A \oplus H) + 1(D \oplus G) \\
(E \oplus L) &= 0(E \oplus L) + 1(E \oplus L) \\
(A \oplus H) &= 0(D \oplus J) + 1(E \oplus L) \\
(D \oplus J) &= 0(A \oplus F) + 1(D \oplus K) \\
(D \oplus K) &= 0(A \oplus H) + 1(D \oplus K)
\end{aligned}
\tag{23}
$$

Since no equation in (23) contains λ, we conclude that $R = S$.

Finally, we note that in a computer implementation of this prodecure, it is not necessary to store all the D-equations of $R \oplus S$. Once all the equations corresponding to new unknowns which appeared on the right-hand side of some preceding D-equation have been generated, the original equation may be erased. If λ appears in any D-equation of $R \oplus S$, the procedure may be terminated immediately with the conclusion that $R \neq S$.

VI. EXTENDED REGULAR EXPRESSIONS

The technique just described for obtaining the D-equations of $R \oplus S$ is applicable to $f(R,S)$ where f is any binary Boolean operator and R and S are restricted regular expressions. The D-equations for the various binary operators differ only in the occurrence or non-occurrence of λ as determined by $f(\delta(D_t R), \delta(D_t S))$.

In the case of the unary operator $\sim$ (negation), the D-equations for $\sim R$ can be found by first deriving the D-equations of

R in the usual manner. Then λ is eliminated from all equations in which it occurs and is added to all equations which did not originally contain λ. The resulting equations are the D-equations of $\sim R$. This follows from the fact that if $\lambda \in R$, then $\lambda \notin \sim R$. Hence $\delta(R) = \lambda$ but $\delta(\sim R) = \phi$. Similarly if $\delta(R) = \phi$, then $\delta(\sim R) = \lambda$. Each X_i which originally denoted some $D_t R$ now denotes $D_t(\sim R)$.

Thus as we implicitly defined binary functions of two sets of D-equations, we can also define the negation (complement) of a set of D-equations. The significant point here is the interpretation of $\delta(X_i)$: the "negation" of λ is ϕ and the "negation" of ϕ is λ. The negation case is best understood by considering state diagrams. A machine which accepts R will accept $\sim R$ if all accepting states are changed to non-accepting states and vice versa. This is analogous to replacing λ by ϕ and ϕ by λ in the D-equations of R.

We now turn to the problem of checking the equivalence of extended regular expressions. Every extended regular expression R may be regarded as a composite Boolean function F of restricted regular expressions $R_1, R_2, \ldots, R_n$. This is denoted by

$$R = F(R_1, R_2, \ldots, R_n) = F(R_i)$$

First, the D-equations of each restricted expression R_i are separately determined; this results in n sets of D-equations. Then by combining the n equations corresponding to $R_1, R_2, \ldots, R_n$, the following composite equation is obtained:

$$F(R_i) = a_1(D_{a_1} F(R_i)) + \ldots + a_k(D_{a_k} F(R_i)) + \delta(F(R_i))$$

This gives rise to further equations just as in the case of $f(R_1,R_2)$. The difference is that n equations must be combined at each stage instead of two. Also $\delta(D_t F(R_i))$ is, in general, more difficult to evaluate than $\delta(D_t f(R_1,R_2))$. $\delta(D_t F(R_i))$ which is a function of λ and ϕ only, is evaluated by applying the normal rules of Boolean algebra with the exception of the special interpretation of $\sim$. As usual, the identity of unknowns is unimportant and they may be replaced by single symbols for brevity.

The following simple example illustrates the method.

Example 4: Suppose $R = \sim(01^* \;\&\; 0^*1)$ where & denotes intersection. We can write this as $\sim(S \;\&\; T)$ where $S = 01^*$ and $T = 0^*1$. The D-equations of S and T are

$$\begin{aligned} S &= 0A + 1B & \qquad T &= 0T + 1C \\ A &= 0B + 1A + \lambda & \qquad C &= 0D + 1D + \lambda \\ B &= 0B + 1B & \qquad D &= 0D + 1D \end{aligned} \tag{24}$$

These two sets of equations are now combined to form the D-equations of $\sim(S \;\&\; T)$.

$$\begin{aligned} \sim(S \,\&\, T) &= 0(\sim(A \,\&\, T)) + 1(\sim(B \,\&\, C)) + \sim(\phi \,\&\, \phi) \\ \sim(A \,\&\, T) &= 0(\sim(B \,\&\, T)) + 1(\sim(A \,\&\, C)) + \sim(\lambda \,\&\, \phi) \\ \sim(B \,\&\, C) &= 0(\sim(B \,\&\, D)) + 1(\sim(B \,\&\, D)) + \sim(\phi \,\&\, \lambda) \\ \sim(B \,\&\, T) &= 0(\sim(B \,\&\, T)) + 1(\sim(B \,\&\, C)) + \sim(\phi \,\&\, \phi) \\ \sim(A \,\&\, C) &= 0(\sim(B \,\&\, D)) + 1(\sim(A \,\&\, D)) + \sim(\lambda \,\&\, \lambda) \\ \sim(B \,\&\, D) &= 0(\sim(B \,\&\, D)) + 1(\sim(B \,\&\, D)) + \sim(\phi \,\&\, \phi) \\ \sim(A \,\&\, D) &= 0(\sim(B \,\&\, D)) + 1(\sim(A \,\&\, D)) + \sim(\lambda \,\&\, \phi) \end{aligned} \tag{25}$$

Evaluating the δ-terms and renaming the unknowns, we get

$$\begin{aligned} R = A &= 0B + 1C + \lambda \\ B &= 0D + 1E + \lambda \\ C &= 0F + 1F + \lambda \\ D &= 0D + 1C + \lambda \\ E &= 0F + 1G \\ F &= 0F + 1F + \lambda \\ G &= 0F + 1G + \lambda \end{aligned} \tag{26}$$

which are the D-equations for R. Now R can be compared with any other regular expression R' by constructing the D-equations of $R \oplus R'$ using (26) and the D-equations of R'. Thus if

$$R' = (00 + 1 + 01(0 + 1))(0 + 1)^* + 0 + \lambda,$$

it can be easily shown that $R = R'$.

VII. MINIMIZATION OF THE D-EQUATIONS

As remarked earlier, a set of D-equations may not be minimal, i.e., two or more unknowns may denote equivalent regular expressions. This may result, for example, from using different symbols to denote equivalent sub-expressions while generating the T-equations.

The problem of minimizing the D-equations is analogous to that of minimizing the states of the corresponding finite-state machine. The latter problem has been extensively studied [5,6]. States are generally distinguished by their responses to various input sequences. This requires some type of iterative table look-up, e.g., inspections of the state table. A modification of this approach is used here to minimize the D-equations. This leads to another method for checking the equivalence of regular expressions.

The set of unknowns U contains all the characteristic derivatives of R; U is therefore called a complete set of derivatives of R. $X, Y \in U$ are said to be λ-equivalent if $\delta(X) = \delta(Y)$. The partition induced on U by λ-equivalence is called a λ-partition. If X and Y are λ-equivalent, we write, using the notation of Hartmanis and Stearns [6], $X \equiv Y(\lambda)$.

Theorem 2: Two elements X and Y in U, the set of unknowns in the D-equations of R, denote equivalent regular expressions if and only if $D_tX \equiv D_tY(\lambda)$ for all $t \in Z^*$.
No proof is needed for this theorem as it is essentially a reformulation of Theorem 1.

The concept of a partition of states with the substitution property [6] can usefully be applied to regular expressions.
Definition: A partition π on a complete set $X_0, X_1, \ldots, X_n$ of characteristic derivatives of a regular expression R is said to have the substitution property if and only if $X_i \equiv X_j(\pi)$ implies that $D_sX_i \equiv D_sX_j(\pi)$ for all $s \in Z^*$.

The minimization procedure is now described. First, the D-equations of R are obtained in the manner already described. This yields a set of unknowns U. Next, partition U under λ-equivalence. We can write this as

$$U = \{\overline{X_1, \ldots, X_h}; \overline{X_k, \ldots, X_n}\} \tag{27}$$

By inspecting the D-equations, find the derivatives of each $X_i \varepsilon U$ with respect to all $a \varepsilon Z$. If $X_i \equiv X_j(\lambda)$, then from Theorem 2 we can say that $X_i \neq X_j$ if

$$D_a X_i \not\equiv D_a X_j(\lambda) \tag{28}$$

for some $a \varepsilon Z$, i.e., if $D_a X_i$ and $D_a X_j$ are in different blocks in (27). Define a new partition U_a^1 in which unknowns in the same block in (27) which do not satisfy (28) are put in separate blocks. Thus U_a^1 is a refinement of U. This operation is repeated for all $a \varepsilon Z$. Let U^1 denote the product $\prod_{a \varepsilon Z} U_a^1$. This partition shows all the elements of U which are distinguished by their derivatives with respect to a single alphabet symbol. Elements in different blocks of U^1 are not equivalent.

The foregoing procedure is repeated, this time unknowns in the same block in U^1 are compared. Unknowns are distinguishable (non-equivalent) if any corresponding one-symbol derivatives belong to different blocks in U^1. In this way, a sequence of partitions, $U, U^1, \ldots U^i$ is produced. The process terminates when two successive partitions are identical, i.e., when

$$U^i = U^{i+1}$$

Derivatives in the same block in U^i are equivalent, derivatives not in the same block are not equivalent. Clearly U^i is a partition with the substitution property.

Thus U has been partitioned into classes of equivalent unknowns. All but one unknown in each equivalence class is eliminated. The equations of the retained unknowns constitute the minimal set of D-equations. This reduced set of D-equations is isomorphic to the set of characteristic equations of the corresponding regular expression. If the sets of reduced D-equations obtained from R and S are isomorphic, i.e., if there is a one-to-one correspondence between the symbols for the unknowns, then $R = S$; otherwise $R \neq S$.

Returning again to Example 3, the set of unknowns U from (22) is $\{F,G,H,J,K,L\}$. Partition U under λ-equivalence.

$$U = \{\overline{F,H,L};\ \overline{G,J,K}\}$$

We proceed to generate refinements of U using (22).

$$U_0^1 = \{\overline{F,H};\ \overline{L};\ \overline{G,J,K}\}$$

$$U_1^1 = \{\overline{F,H,L};\ \overline{G,J,K}\}$$

Hence

$$U^1 = U_0^1 \cdot U_1^1 = \{\overline{F,H};\ \overline{L};\ \overline{G,J,K}\} \tag{29}$$

(29) is now refined by comparing the first-order derivatives of unknowns in the same block. The result is

$$U^2 = \{\overline{F,H};\ \overline{L};\ \overline{G,J,K}\}$$

which is identical to U^1. It follows that $F = H$ and $G = J = K$.

Equations (21) now reduce to

$$\begin{aligned} S = F &= 0G + 1L \\ G &= 0F + 1G + \lambda \\ L &= 0L + 1L \end{aligned} \tag{30}$$

Comparing S to R in Example 1, it is seen that (30) and (15) are isomorphic (where $A \rightarrow F$, $D \rightarrow G$ and $F \rightarrow L$). Hence we again conclude that R and S are equivalent.

It is quite apparent that much less effort is needed to construct the D-equations of $R \oplus S$ than to minimize the D-equations of R and S. The former approach is therefore more suitable for a practical algorithm to determine equivalence. However, the latter approach is significant in that it makes it possible to reduce all regular expressions to canonical forms.

Consider the minimized D-equations (30) obtained from S in Example 3. These may be solved for S by eliminating all other unknowns by means of Arden's Theorem and substitution. Application of Arden's Theorem to the equation for L yields

$$L = (0 + 1)^*\phi = \phi$$

Similarly for G,

$$G = 1^*(0F + \lambda)$$

Substitute for L and G in the equation for S.

$$\begin{aligned} S = F &= 01^*0F + 01^* \\ &= (01^*0)^*01^* \end{aligned}$$

This is equivalent to but much simpler than the original expression for S. It is in fact identical to R in Example 1.

The structure of the regular expression obtained by solving the minimized D-equations depends on the order in which unknowns are eliminated. By specifying this order, (e.g., eliminate the unknown X_i with the highest subscript first, then X_{i-1}, etc.) the form of the resultant regular expression is fixed. This means that all equivalent expressions can be reduced to a unique form. This canonical form is always a restricted regular expression.

VIII. SUMMARY AND CONCLUSIONS

Some existing approaches to the problem of checking regular expressions for equivalence were considered and their limitations were noted. A practical checking algorithm based on the construction of regular equations was described. Two types of equation were used: the T-equations which are closely related to transition graphs, and the D-equations which are related to (deterministic) state graphs and state tables. The D-equations of R and S are used to construct D-equations for $R \oplus S$, from which equivalence may be determined by inspection. This technique is applicable to both restricted form and extended regular expressions.

Bounds were obtained for the number of equations which may be generated. The procedure, which essentially involves algebraic symbol manipulation, is particularly suited to the comparison of long regular expressions and to implementation on a computer.

In addition, a method for minimizing the sets of D-equations was outlined. It was shown that the equivalence of two regular expressions can be determined by comparing their respective minimized D-equations. This technique is analogous to finding and comparing the minimum-state machines which accept the given expressions. Finally, it was shown that the solution of the minimized D-equations leads to a canonical form for equivalent expressions.

In dealing with very long regular expressions, it might be worthwhile to combine the two checking algorithms. Minimization (or even partial reduction) of intermediate sets of D-equations may reduce the total amount of computation required. Redundancy in a regular expression tends to introduce a corresponding redundancy in the T-equations and ultimately in the D-equations.

This investigation has shown the usefulness of regular equations for checking equivalence. Furthermore, regular equations provide a valuable tool for other operations with regular expressions, e.g., conversion to restricted form and length reduction.

ACKNOWLEDGEMENT: The author wishes to express his gratitude to Mr. Sharad Seth and Professor Gernot Metze for their helpful criticism.

REFERENCES

1. GINZBURG, A. A Procedure for checking the equality of regular expressions. J. ACM 14 (April 1967), 355-362.

2. BRZOZOWSKI, J. A. Review of Reference [1] in CR 8 (Sept-Oct 1967), 474.

3. BRZOZOWSKI, J. A. Derivatives of regular expressions. J. ACM 11 (Jan 1964), 481-494.

4. KUCK, D. Lecture notes for CS 301, University of Illinois, Urbana, Illinois, Spring 1967.

5. HARRISON, M. A. Introduction to Switching and Automata Theory, McGraw-Hill, New York, 1965.

6. HARTMANIS, J. and STEARNS, R. E. Algebraic Structure Theory of Sequential Machines, Prentice-Hall, Englewood Cliffs, N. J., 1966.

DISTRIBUTION LIST AS OF APRIL 1, 1967

1 Dr. Edward M. Reilley
Asst. Director (Research)
Ofc. of Defense Res. & Engrg.
Department of Defense
Washington, D. C. 20301

1 Office of Deputy Director
(Research and Information Rm. 3D1037)
Department of Defense
The Pentagon
Washington, D. C. 20301

1 Director
Advanced Research Projects Agency
Department of Defense
Washington, D. C. 20301

1 Director for Materials Sciences
Advanced Research Projects Agency
Department of Defense
Washington, D. C. 20301

1 Headquarters
Defense Communications Agency (333)
The Pentagon
Washington, D. C. 20305

50 Defense Documentation Center
Attn: TISIA
Cameron Station, Bldg. 5
Alexandria, Virginia 22314

1 Director
National Security Agency
Attn: TDL
Fort George G. Meade, Maryland 20755

1 Weapons Systems Evaluation Group
Attn: Col. Daniel W. McElwee
Department of Defense
Washington, D. C. 20305

1 National Security Agency
Attn: R4-James Tippet
Office of Research
Fort George G. Meade, Maryland 20755

1 Central Intelligence Agency
Attn: OCR/DD Publications
Washington, D. C. 20505

1 Colonel Kee
AFRSTE
Hqs. USAF
Room 1D-429, The Pentagon
Washington, D. C. 20330

1 Colonel A. Swan
Aerospace Medical Division
Brooks Air Force Base, Texas 78235

1 AUL3T-9663
Maxwell AFB, Alabama 36112

1 AFFTC (FTBPP-2)
Technical Library
Edwards AFB, California 93523

1 Space Systems Division
Air Force Systems Command
Los Angeles Air Force Station
Los Angeles, California 90045
Attn: SSSD

1 Major Charles Waespy
Technical Division
Deputy for Technology
Space Systems Division, AFSC
Los Angeles, California 90045

1 SSD(SSTRT/Lt. Starbuck)
AFUPO
Los Angeles, California 90045

1 Det. #6, OAR (LOOAR)
Air Force Unit Post Office
Los Angeles, California 90045

1 Systems Engineering Group (RTD)
Technical Information Reference Branch
Attn: SEPIR
Directorate of Engineering Standards
& Technical Information
Wright-Patterson AFB, Ohio 45433

1 ARL (ARIY)
Wright-Patterson AFB, Ohio 45433

1 Dr. H. V. Noble
Air Force Avionics Laboratory
Wright-Patterson AFB, Ohio 45433

1 Mr. Peter Murray
Air Force Avionics Laboratory
Wright-Patterson AFB, Ohio 45433

1 AFAL (AVTE/R.D. Larson)
Wright-Patterson AFB, Ohio 45433

2 Commanding General
Attn: STEWS-WS-VT
White Sands Missile Range,
New Mexico 88002

1 RADC (EMLAL-I)
Griffiss AFB, New York 13442
Attn: Documents Library

1 Academy Library (DFSLB)
U. S. Air Force Academy
Colorado Springs, Colorado 80912

1 Lt. Col. Bernard S. Morgan
Frank J. Seiler Research Laboratory
U. S. Air Force Academy
Colorado Springs, Colorado 80912

1 APGC (PGBPS-12)
Elgin AFB, Florida 32542

1 Commanding Officer
Human Engineering Laboratories
Aberdeen Proving Ground, Maryland 21005

1 Director
U. S. Army Engineer Geodesy, Intelligence
and Mapping
Research and Development Agency
Fort Belvior, Virginia 22060

1 Commandant
U. S. Army Command and General Staff College
Attn: Secretary
Fort Leavenworth, Kansas 66270

1 Dr. H. Robl
Deputy Chief Scientist
U. S. Army Research Office (Durham)
Box CM, Duke Station
Durham, North Carolina 27706

1 Commanding Officer
U. S. Army Research Office (Durham)
Attn: CRD-AA-IP (Richard O. Ulsh)
Box CM, Duke Station
Durham, North Carolina 27706

1 Librarian
U. S. Army Military Academy
West Point, New York 10996

1 The Walter Reed Institute of Research
Walter Reed Medical Center
Washington, D. C. 20012

1 Commanding Officer
U. S. Army Electronics R&D Activity
Fort Huachuca, Arizona 85163

1 Commanding Officer
U. S. Army Engineer R&D Laboratory
Attn: STINFO Branch
Fort Belvoir, Virginia 22060

1 Commanding Officer
U. S. Army Electronics R&D Activity
White Sands Missile Range, New Mexico 88002

1 Dr. S. Benedict Levin, Director
Institute for Exploratory Research
U. S. Army Electronics Command
Fort Monmouth, New Jersey 07703

1 Director
Institute for Exploratory Research
U. S. Army Electronics Command
Attn: Mr. Robert O. Parker, Executive
Secretary, JSTAC (AMSEL-XL-D)
Fort Monmouth, New Jersey 07703

1 Commanding General
U. S. Army Electronics Command
Fort Monmouth, New Jersey 07703

Attn: AMSEL-SC
RD-D
RD-G
RD-GF
RD-MAT
XL-D
XL-E
XL-C
XL-S
HL-D
HL-CT-R
HL-CT-P
HL-CT-L
HL-CT-O
HL-CT-I
HL-CT-A
NL-D
NL-A
NL-P
NL-R
NL-S
KL-D
KL-E
KL-S
KL-T
VL-D
WL-D

1 Chief of Naval Research
Department of the Navy
Washington, D. C. 20360
Attn: Code 427

3 Chief of Naval Research
Department of the Navy
Washington, D. C. 20360
Attn: Code 437

2 Naval Electronics Systems Command
ELEX 03
Falls Church, Virginia 22046

1 Naval Ship Systems Command
SHIP 031
Washington, D. C. 20360

1 Naval Ship Systems Command
SHIP 035
Washington, D. C. 20360

2 Naval Ordnance Systems Command
ORD 32
Washington, D. C. 20360

2 Naval Air Systems Command
AIR 03
Washington, D. C. 20360

2 Commanding Officer
Office of Naval Research Branch Office
Box 39, Navy No. 100 F.P.O.
New York, New York 09510

1 AFETR Technical Library
(ETV, MU-135)
Patrick AFB, Florida 32925

1 AFETR (ETLLG-I)
STINFO Officer (For Library)
Patrick AFB, Florida 32925

1 Dr. L. M. Hollingsworth
AFCRL (CRN)
L. G. Hanscom Field
Bedford, Massachusetts 01731

1 AFCRL (CRMXLR)
AFCRL Research Library, Stop 29
L. G. Hanscom Field
Bedford, Massachusetts 01731

1 Colonel Robert E. Fontana
Department of Electrical Engineering
Air Force Institute of Technology
Wright-Patterson AFB, Ohio 45433

1 Colonel A. D. Blue
RTD (RTTL)
Bolling Air Force Base, D. C. 20332

1 Dr. I. R. Mirman
AFSC (SCT)
Andrews AFB, Maryland 20331

1 Colonel J. D. Warthman
AFSC (SCTR)
Andrews AFB, Maryland 20331

1 Lt. Col. J. L. Reeves
AFSC (SCBB)
Andrews AFB, Maryland 20331

2 ESD (ESTI)
L. G. Hanscom Field
Bedford, Massachusetts 01731

1 AEDC (ARO, INC)
Attn: Library/Documents
Arnold AFS, Tennessee 37389

2 European Office of Aerospace Research
Shell Building
47 Rue Cantersteen
Brussels, Belgium

5 Lt. Col. Robert B. Kalisch
Chief, Electronics Division
Directorate of Engineering Sciences
Air Force Office of Scientific Research
Arlington, Virginia 22209

1 U. S. Army Research Office
Attn: Physical Sciences Division
3045 Columbia Pike
Arlington, Virginia 22204

1 Research Plans Office
U. S. Army Research Office
3045 Columbia Pike
Arlington, Virginia 22204

1 Commanding General
U. S. Army Materiel Command
Attn: AMCRD-RS-DE-E
Washington, D. C. 20315

1 Commanding General
U. S. Army Strategic Communications Command
Washington, D. C. 20315

1 Commanding Officer
U. S. Army Materials Research Agency
Watertown Arsenal
Watertown, Massachusetts 02172

1 Commanding Officer
U. S. Army Ballistics Research Laboratory
Attn: V. W. Richards
Aberdeen Proving Ground
Aberdeen, Maryland 21005

1 Commandant
U. S. Army Air Defense School
Attn: Missile Sciences Division, C&S Dept.
P.O. Box 9390
Fort Bliss, Texas 79916

1 Redstone Scientific Information Center
Attn: Chief, Document Section
Redstone Arsenal, Alabama 35809

1 Commanding General
Frankford Arsenal
Attn: SMUFA-1310 (Dr. Sidney Ross)
Philadelphia, Pennsylvania 19137

1 U. S. Army Munitions Command
Attn: Technical Information Branch
Picatinney Arsenal
Dover, New Jersey 07801

1 Commanding Officer
Harry Diamond Laboratories
Attn: Dr. Berthold Altman (AMXDO-TI)
Connecticut Avenue and Van Ness Street, N.W.
Washington, D. C. 20438

1 Commanding Officer
U. S. Army Security Agency
Arlington Hall
Arlington, Virginia 22212

1 Commanding Officer
U. S. Army Limited War Laboratory
Attn: Technical Director
Aberdeen Proving Ground
Aberdeen, Maryland 21005

1 Commanding Officer
Office of Naval Research Branch Office
219 South Dearborn Street
Chicago, Illinois 60604

1 Commanding Officer
Office of Naval Research Branch Office
1030 East Green Street
Pasadena, California 91101

1 Commanding Officer
Office of Naval Research Branch Office
207 West 24th Street
New York, New York 10011

1 Commanding Officer
Office of Naval Research Branch Office
495 Summer Street
Boston, Massachusetts 02210

8 Director, Naval Research Laboratory
Technical Information Officer
Washington, D. C. 20390
Attn: Code 2000

1 Commander
Naval Air Development and Material Center
Johnsville, Pennsylvania 18974

2 Librarian
U. S. Naval Electronics Laboratory
San Diego, California 95152

1 Commanding Officer and Director
U. S. Naval Underwater Sound Laboratory
Fort Trumbull
New London, Connecticut 06840

1 Librarian
U. S. Navy Post Graduate School
Monterey, California 93940

1 Commander
U. S. Naval Air Missile Test Center
Point Mugu, California 95468

1 Director
U. S. Naval Observatory
Washington, D. C. 20390

2 Chief of Naval Operations
OP-07
Washington, D. C. 20350

1 Director, U. S. Naval Security Group
Attn: G43
3801 Nebraska Avenue
Washington, D. C. 20016

2 Commanding Officer
Naval Ordnance Laboratory
White Oak, Maryland 21162

1 Commanding Officer
Naval Ordnance Laboratory
Corona, California 91720

1 Commanding Officer
Naval Ordnance Test Station
China Lake, California 93555

1 Commanding Officer
Naval Avionics Facility
Indianapolis, Indiana 46218

1 Commanding Officer
Naval Training Device Center
Orlando, Florida 32813

1 U. S. Naval Weapons Laboratory
Dahlgren, Virginia 22448

1 Weapons Systems Test Division
Naval Air Test Center
Patuxtent River, Maryland 20670
Attn: Library

1 Head, Technical Division
U. S. Naval Counter Intelligence Support Center
Fairmont Building
4420 North Fairfax Drive
Arlington, Virginia 22203

1 Mr. Charles F. Yost
Special Asst. to the Director of Research
National Aeronautics and Space Administration
Washington, D. C. 20546

1 Dr. H. Harrison, Code RRE
Chief, Electrophysics Branch
National Aeronautics and Space Administration
Washington, D. C. 20546

1 Goddard Space Flight Center
National Aeronautics and Space Administration
Attn: Library C3/TDL
Green Belt, Maryland 20771

1 NASA Lewis Research Center
Attn: Library
21000 Brookpark Road
Cleveland, Ohio 44135

1 National Science Foundation
Attn: Dr. John R. Lehmann
Division of Engineering
1800 G Street, N.W.
Washington, D. C. 20550

1 U. S. Atomic Energy Commission
Division of Technical Information Extension
P. O. Box 62
Oak Ridge, Tennessee 37831

1 Los Alamos Scientific Laboratory
Attn: Reports Library
P. O. Box 1663
Los Alamos, New Mexico 87544

2 NASA Scientific & Technical Information Facility
Attn: Acquisitions Branch (S/AK/DL)
P. O. Box 33
College Park, Maryland 20740

1 Director
Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

1 Polytechnic Institute of Brooklyn
55 Johnson Street
Brooklyn, New York 11201
Attn: Mr. Jerome Fox
Research Coordinator

1 Director
Columbia Radiation Laboratory
Columbia University
538 West 120th Street
New York, New York 10027

1 Director
Coordinated Science Laboratory
University of Illinois
Urbana, Illinois 61801

1 Director
Stanford Electronics Laboratories
Stanford University
Stanford, California 94305

1 Director
Electronics Research Laboratory
University of California
Berkeley, California 94720

1 Director
Electronic Sciences Laboratory
University of Southern California
Los Angeles, California 90007

1 Professor A. A. Dougal, Director
Laboratories for Electronics and Related Sciences Research
University of Texas
Austin, Texas 78712

1 Division of Engineering and Applied Physics
210 Pierce Hall
Harvard University
Cambridge, Massachusetts 02138

1 Aerospace Corporation
P. O. Box 95085
Los Angeles, California 90045
Attn: Library Acquisitions Group

1 Professor Nicholas George
California Institute of Technology
Pasadena, California 91109

1 Aeronautics Library
Graduate Aeronautical Laboratories
California Institute of Technology
1201 East California Boulevard
Pasadena, California 91109

1 Director, USAF Project RAND
Via: Air Force Liaison Office
The RAND Corporation
1700 Main Street
Santa Monica, California 90406
Attn: Library

1 The Johns Hopkins University
Applied Physics Laboratory
8621 Georgia Avenue
Silver Spring, Maryland 20910
Attn: Boris W. Kuvshinoff
Document Librarian

1 Hunt Library
Carnegie Institute of Technology
Schenley Park
Pittsburgh, Pennsylvania 15213

1 Dr. Leo Young
Stanford Research Institute
Menlo Park, California 94025

1 Mr. Henry L. Bachmann
Assistant Chief Engineer
Wheeler Laboratories
122 Cuttermill Road
Great Neck, New York 11021

1 School of Engineering Sciences
Arizona State University
Tempe, Arizona 85281

1 University of California at Los Angeles
Department of Engineering
Los Angeles, California 90024

1 California Institute of Technology
Pasadena, California 91109
Attn: Documents Library

1 University of California
Santa Barbara, California 93106
Attn: Library

1 Carnegie Institute of Technology
Electrical Engineering Department
Pittsburgh, Pennsylvania 15213

1 University of Michigan
Electrical Engineering Department
Ann Arbor, Michigan 48104

1 New York University
College of Engineering
New York, New York 10019

1 Syracuse University
Department of Electrical Engineering
Syracuse, New York 13210

1 Yale University
Engineering Department
New Haven, Connecticut 06520

1 Airborne Instruments Laboratory
Deerpark, New York 11729

1 Bendix Pacific Division
11600 Sherman Way
North Hollywood, California 91605

1 General Electric Company
Research Laboratories
Schenectady, New York 12301

1 Lockheed Aircraft Corporation
P. O. Box 504
Sunnyvale, California 94088

1 Raytheon Company
Bedford, Massachusetts 01730
Attn: Librarian

1 Dr. G. J. Murphy
The Technological Institute
Northwestern University
Evanston, Illinois 60201

1 Dr. John C. Hancock, Director
Electronic Systems Research Laboratory
Purdue University
Lafayette, Indiana 47907

1 Director
Microwave Laboratory
Stanford University
Stanford, California 94305

1 Emil Schafer, Head
Electronics Properties Info Center
Hughes Aircraft Company
Culver City, California 90230

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)* University of Illinois, Coordinated Science Laboratory, Urbana, Illinois 61801	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP

3. REPORT TITLE

ON THE EQUILVALENCE OF REGULAR EXPRESSIONS

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

5. AUTHOR(S) *(First name, middle initial, last name)*

HAYES, JOHN P.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
April 1968	25	6

8a. CONTRACT OR GRANT NO. DAAB-07-67-C-0199; also in part NSF GK-1663	9a. ORIGINATOR'S REPORT NUMBER(S) R-374
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT

Distribution of this report is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Joint Services Electronics Program thru U.S. Army Electronics Command Ft. Monmouth, New Jersey 07703

13. ABSTRACT

Some existing techniques for checking the equivalence of regular expressions are examined and their shortcomings discussed. A checking procedure based on the construction of regular equations is described. First, a set of transition or T-equations is obtained from the given expressions R and S. These are then converted to derivative or D-equations. Finally, a set of D-equations corresponding to $R \oplus S$ is constructed. It is shon that R = S if and only if the D-equations of $R \oplus S$ do not contain the empty string λ.

Upper bounds are determined for the number of equations involved in each step of the procedure. Unlike previously-described methods, neither the construction of the graphs nor the computation of derivatives is required. This technique is also applicable to extended regular expressions involving any Boolean operators.

A modified procedure involving minimization of the D-equations of R and S is also discussed. R = S if and only if their respective minimized D-eauations are isomorphic. Finally, it is shown that these equations lead to a canonical form for equivalent expressions.

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Regular Expressions						
Equivalence						
Regular Equations						
Derivatives of regular expressions						
Extended regular expressions						
Transition Graphs						
Finite-state machines						
Minimization						
Canonical Forms						