A NEW ALGORITHM FOR SYMBOLIC RELIABILITY ANALYSIS OF COMPUTER - COMMUNICATION NETWORKS[*]

A. Grnarov[**], L. Kleinrock and M. Gerla
Computer Science Department
University of California
Los Angeles, California

ABSTRACT

A new algorithm for symbolic network analysis is presented. The algorithm is based on the application of a newly defined \$-operation on the set of all simple paths.

Comparisons with existing algorithms on the basis of terms that must be evaluated during the derivation, the number of operations required, and the execution time in several represented benchmarks show that the proposed algorithm is considerably more efficient than currently available solutions.

I. INTRODUCTION

The symbolic analysis of reliability networks has been the subject of considerable research. The symbolic terminal reliability algorithms available in the literature are based on path enumeration [1-13], cut-set enumeration [4,8] and on the acyclic subgraphs of the given probabilistic graph [14].

The most efficient path enumeration algorithms use reduction to mutually exclusive events by Boolean algebra. However, common drawbacks of these algorithms are that they generate a large number of terms [3, 13]; that they cannot efficiently handle systems of medium to large size (i.e., system graphs having more than 20 paths between input-output node pairs) [4]; and that they cannot easily determine the resulting symbolic function when the number of elements in the network is large [8]. The same drawbacks affect the algorithms based on the cut-set enumeration [4].

A more efficient algorithm based on the acyclic subgraphs of the given probabilistic graph was recently proposed by Satyanarayana and Prabhakar [14]. The examples indicate that this algorithm is appreciable faster than existing methods and can handle larger networks.

In this paper a new algorithm for the symbolic terminal reliability computation is presented. The algorithm belongs to the class of path enumeration algorithms (which use Boolean algebra) It is based on the application of a newly defined \$-operation on the set of all simple paths. A simple path is represented by a binary string (path identifier); thus only logical operations are required. The algorithm does not suffer from the drawbacks of path or cut-set enumeration algorithms.

[*]This research was supported in part by the Advanced Research Projects Agency of the Department of Defense under contract MDA 903-77-0272 and in part by the Office of Naval Research under contract N00014-79-C-0866.

[**]Dr Grnarov is currently on leave from the Electrical Engineering Department, University of Skopje, Yugoslavia.

The algorithm was coded in FORTRAN IV and run on a DEC-10 timesharing computer system. Execution times confirm the advantages of the proposed algorithm over existing path and cut-set enumeration algorithms. The execution times are considerably shorter than the times shown for the network examples in [14] (these results, of course, provide only a qualitative comparison of computational efficiency since the programs were run on different computers).

II. DERIVATION OF THE ALGORITHM

For a network consisting of N nodes and E links, the path identifier is introduced by the following:

Definition 1: The path identifier IP_k for the path π_k is defined as a string of n binary variables

$$IP_k = x_1 x_2 \cdots x_i \cdots x_n$$

where

$x_i = 1$ if the i^{th} element of the network (node or link) is included in the path π_k

$x_i = x$ otherwise

and n = number of network elements that can fail, i.e.:

n = N in the case of perfect links and imperfect nodes

n = E in the case of perfect nodes and imperfect links

n = N + E in the case of imperfect links and imperfect nodes

As an example let us consider a 4 node, 5 link network, given in Figure 1, in which nodes are perfectly reliable and links are subject to failures. The path (s,x_1,a,x_5,b,x_4,t) is then represented by the string: IP = 1xx11.

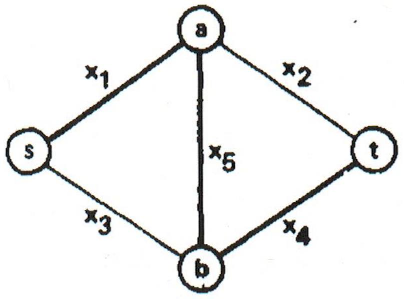

Figure 1. Path Identifier String

According to *Definition 1*, it can be seen that a path identifier has the form of the cube in Boolean algebra. Hence, by applying the #-operation ("sharp" operation) ([15], page 173) between two path identifiers, denoted by $IP_k \# IP_j$, we obtain all subcubes (events) of IP_k not included in IP_j. Unfortunately, the generated subcubes are not mutually disjoint. The #-operation should be repeated

Since the proposed algorithm SYMB does not produce cancelling terms, the obtained result for the example given in Figure 1, has four terms in comparison with [4]and [3] which have five and six terms respectively.

For the second example of a network, shown in Figure 2 (also Figures 1, 10 and 3 in [8], [4] and [14] respectively), we obtain a reliability expression which has 16 terms. The results in [8]a and [4] have 22 and 61 terms respectively, and the symbolic expression in [14] upon expansion yields 123 terms.

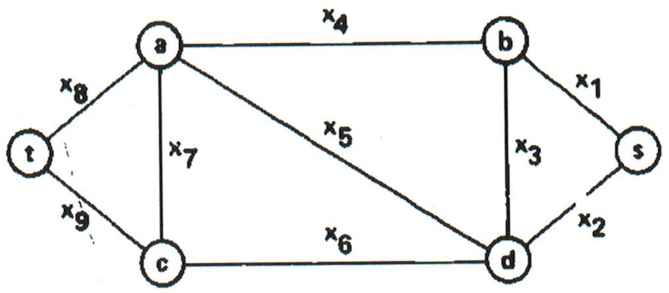

Figure 2. Modified Graph of ARPA Network

The third example, given in Figure 3, was also used in [3], [13] and [12]. The reliability expression has 38 terms while the result in [13] has 72 terms. For the determination of S_{23}, The algorithm SYMB requires 34 comparisons while the algorithms reported in [13] and [12] require 54 and 173 comparisons respectively For finding S_{24} these values are 26, 45 and 159 respectively.

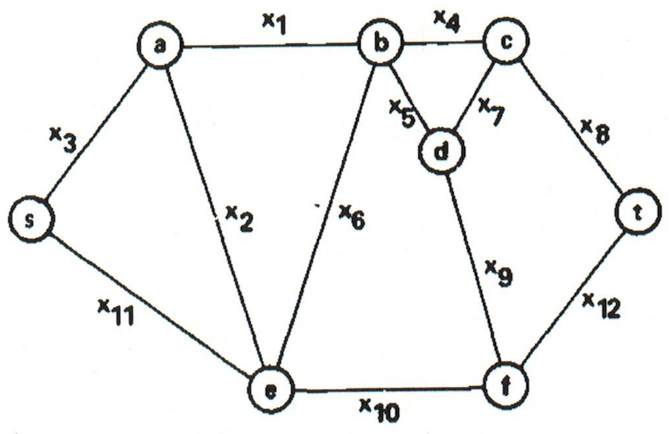

Figure 3. Example Network

A program based on the proposed algorithm was written in FORTRAN IV and run on a DEC-10 timesharing computer system. The program was applied to a number of network examples found in the literature. The execution times for examples given in Figures 2 through 6 are presented in Table 3 and are compared with the execution times of some of the existing algorithms. The results confirm the efficiency of the proposed algorithm.

on the set of subcubes again and again until a set of disjoint subcubes is obtained.

In the case when the paths π_j, $j = 1, 2, \cdots, m-1$, have been examined, the determination of the set of subcubes S_m of the path π_m, which are not included in the previous paths, can be performed as

$$S_m = (...((IP_m \# IP_1) \# IP_2) \# ... \# IP_j) \# ...) \# IP_{m-1} \qquad (1)$$

Since the #-operation is performed using the largest possible cubes (path identifiers), the generation of S_m is faster as compared with [3] and [8]; also there is no need for storing the new terms. The drawbacks of operation (1) are the generation of the repeated subcubes and the need for the repeated application of the #-operation between the subcubes in S_m. To avoid these drawbacks, a new \$-operation is introduced in *Definition 4* below.

Before the presentation of *Definition 4* it is useful to make the following consideration. Since the path identifiers contain only symbols 1 and x, the #-operation can produce as a new symbol, only 0. The 0's generated in step j are designated as 0^j. If in the cube generated by #-operation there is only one 0^j, we call this a unique 0. Next we quote the definition of the coordinate #-operation from [15] (the definition of #-operation between two cubes is given in [15] on page 173) and we introduce the definitions of the $@_j$-operation and the \$-operation.

Definition 2. The coordinate #-operation is defined as given in Table 1.

Table 1

a_i \ b_i	0	1	x
0	z	y	z
1	y	z	z
x	1	0	z

Definition 3. The $@_j$-operation between two cubes, say $C^r = a_1 a_2 \cdots a_i \cdots a_n$ and $C^s = b_1 b_2 \cdots b_i \cdots b_n$, is defined as

$$C^r @_j C^s = \begin{cases} C^r & \text{if } a_i \# b_i = y \text{ for any unique 0 or} \\ & \text{if } a_i \# b_i = y \text{ for all } a_i = 0^v \text{ for any v} \\ C_1^r \# C^s \cup C_0^r & \text{otherwise} \end{cases}$$

where

C_1^r is a cube obtained from C^r substituting all $a_i = 0^j$, for which $a_i \# b_i = y$, by 1.

and

C_0^r is a cube obtained from C^r substituting all $a_i = 0^j$, for which $a_i \# b_i \neq y$, by x.

Definition 4. The \$-operation between two cubes C^r and C^s is defined as

$$C^r \$ C^s = \begin{cases} \Phi & \text{if } a_i \# b_i = z \text{ for all i} \\ S_k & \text{if } a_i \# b_i = y \text{ for any i} \\ C^r & \text{otherwise} \end{cases}$$

where Φ is the empty set. $S_j = S_{j-1} @_j C^s$, $S_1 = C' @_1 C^s$ and $j = 2, 3, \cdots k$ (k is the number of steps in which any symbol(s) $\dot{0}$ is generated), $C' = c_1 c_2 \cdots c_i \cdots c_n$ and $c_i = a_i \# b_i$.

According to *Definition 4* it follows that if we substitute the #-operation in (1) by the \$-operation, the resulting set S_m will consist of disjoint cubes only. The symbolic expression, corresponding to a cube C in S_m, is given by

$$T(C) = P \cdot Q \prod_{j=2}^{k} (1 - P_j)$$

where

$P = \Pi\, p_i$	for all i satisfying $c_i = 1$
$Q = \Pi\, q_i$	for all i satisfying $c_i =$ unique 0
$P_j = \Pi\, p_i$	for all i satisfying $c_i = 0^j$
p_i	is symbol representing probability that the i^{th} element is up
$q_i = 1 - p_i$	

and Π and $\cdot$ are the product and concatenation operators in the string algebra.

We can now introduce the algorithm SYMB for the derivation of symbolic expression for terminal reliability:

ALGORITHM SYMB:

```
step 1.          Find path identifiers for all simple paths between node s and node t
step 2.          Sort them according to increasing number of symbols 1 (i.e., increasing path
                 length).
step 3.          Set $T_{11} = T(IP_1)$
step 4. (loop)   For $m = 2, 3, \cdots, k$ determine
                 $S_m = (\cdots((IP_m \$ IP_1)\$IP_2 \cdots)\$IP_{m-1}$
                 Form $T_{mi} = T(C_m^i)$ , $i = 1, 2, \cdots, l$
step 5.          END
```

In the algorithm, C_m^i is a cube in the set S_m and l is the number of cubes in S_m.

As an example, the SYMB algorithm is applied to the simple bridge network given in Figure 1. (This same example was used in [3]and [4].)

steps 1,2. Table 2 presents the set of all simple paths and their path identifiers (for simplicity we consider the case with perfect nodes and imperfect links).

Table 2

m	π_m	IP_m
1	x_1x_2	11xxx
2	x_3x_4	xx11x
3	$x_1x_5x_4$	1xx11
4	$x_3x_5x_2$	x11x1

step 3. $T_{11} = p_1p_2$

step 4. $m = 2$:

$$\begin{array}{rll} & \text{xx11x} & \\ & \underline{\text{11xxx}} & \\ S_2 = & \text{0011x} & j = 1 \end{array}$$

$T_{21} = p_3p_4(1 - p_1p_2)$

$m = 3$:

$$\begin{array}{rll} & \text{1xx11} & \\ & \underline{\text{11xxx}} & \\ & \text{10x11} & \text{unique 0, } j = 1 \\ & \underline{\text{xx11x}} & \\ S_3 = & \text{10011} & \text{unique 0, } j = 2 \end{array}$$

$T_{31} = p_1p_4p_5q_2q_3$

$m = 4$:

$$\begin{array}{rll} & \text{x11x1} & \\ & \underline{\text{11xxx}} & \\ & \text{011x1} & \text{unique 0, } j = 1 \\ & \underline{\text{xx11x}} & \\ & \text{01101} & \text{unique 0, } j = 2 \\ & \underline{\text{1xx11}} & \\ S_4 = & \text{01101} & \end{array}$$

$T_{41} = p_2p_3p_5q_1q_4$

step 5. END

The symbolic expression for terminal reliability is

$$T_{st} = p_1p_2 + p_3p_4(1 - p_1p_2) + p_1p_4p_5q_2q_3 + p_2p_3p_5q_1q_4$$

III. COMPUTATIONAL RESULTS

Two common criteria for the evaluation of symbolic reliability algorithms are: (1) the number of terms in the reliability expression; and (2) the number of comparisons of an intermediate product term with the term represented by a simple path [13].

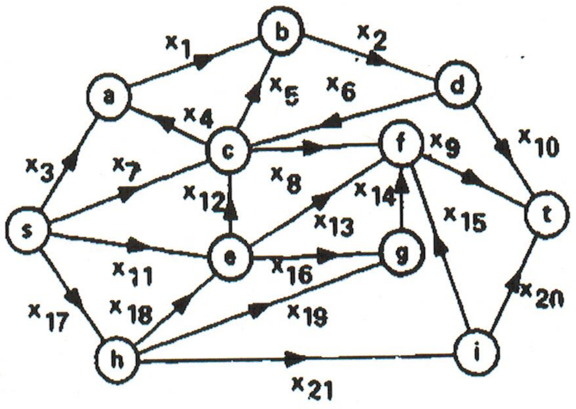

Figure 4. Topology of a Highly Connected Network

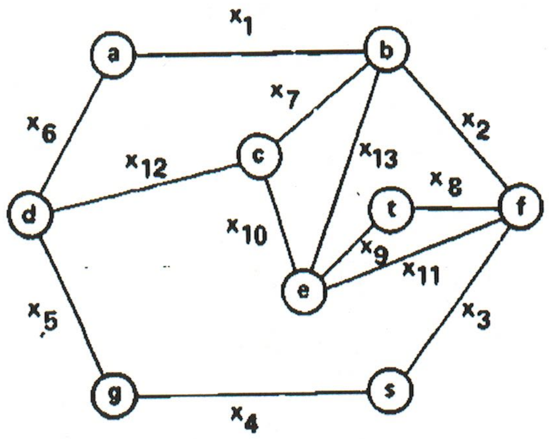

Figure 5. A Long Line Telephone Network Example

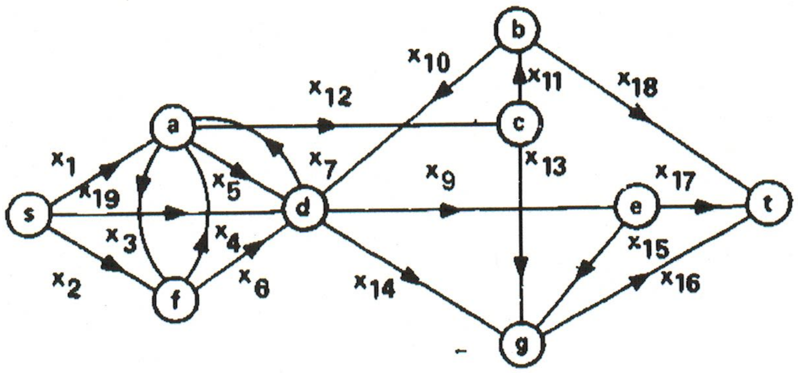

Figure 6. Network Example Which Allows for Both Nodes and Links Failures

Table 3

Algorithm	Computer system/language	Fig. 2	Fig. 3	Fig. 4	Fig. 5	Fig. 6
Fratta, Montanari [3]	IBM 360/67 FORTRAN		112 s.	> 10 min.		
Lin, Leon, Huang [4]	CDC 6500 FORTRAN	9 s.			75 s.	
Abraham [13]	DEC-10 SAIL		6 s.	19 s.		
Satyanarayana, Prabhakar [14]	PDP 11/45 FORTRAN	1.2 s			5.7 s.	43 s.
SYMB	DEC-10 FORTRAN	0.8 s.	1.5 s.	3.3 s.	1.1 s.	7.4 s.

IV. CONCLUDING REMARKS

The paper presents a new algorithm for symbolic network analysis. The proposed algorithm is based on the implementation of a newly defined \$-operation on the set of path identifiers. Applying the \$-operations, only the disjoint subcubes are generated and therefore the reliability expression can be obtained in a straightforward manner. The algorithm is efficient since it does not generate a large number of terms and does not require the generation and storage of intermediate terms beside the path identifiers.

The comparisons with the existing algorithms in the number of terms, number of comparisons and execution times of the realized program confirm the efficiency of the proposed algorithm.

Due to the improved computational efficiency, the SYMB algorithm permits us to analyze and derive symbolic reliability expressions for networks of considerably larger size than was possible using the previous techniques.

REFERENCES

[1] K. Misra, "An algorithm for the reliability evaluation of redundant networks," *IEEE Trans. Reliability*, vol. R-19, November 1970, pp. 146-151.

[2] C. Lee, "Analysis of switching networks," *Bell System Tech. J.*, November 1955, pp. 1287-1315.

[3] L. Fratta, U. Montanari, "A Boolean algebra method for computing the terminal reliability in a communication network," *IEEE Trans. Circuit Theory*, vol. CT-20, May 1973, pp. 203-211.

[4] P. Lin, B. Leon, and T. Huang, "A new algorithm for symbolic system reliability analysis," *IEEE Trans. Reliability*, vol. R-25, no. 1, April 1976, pp. 2-14.

[5] L. Fratta, U. Montanari, "Synthesis of available networks," *IEEE Trans. Reliability*, vol. R-25, no. 2, June 1976, pp. 81-87.

[6] J. de Mercado, N. Spyratos and B. Bowen, "A method for calculation of network reliability," *IEEE Trans. Reliability*, vol. R-25, no. 2, June 1976, pp. 71-76.

[7] P. Canarda, F. Corsi and A. Trentadue, "An efficient simple algorithm for fault free automatic synthesis from the reliability graph," *IEEE Trans. Reliability*, vol. R-27, no. 3, August 1978, pp. 215-221.

[8] S. Rai and K. Aggarwal, "An efficient method for reliability evaluation of a general network," *IEEE Trans. Reliability*, vol. R-27, no. 3, August 1978, pp. 206-211.

[9] T. Case, "A reduction technique for obtaining a simplified reliability expression," *IEEE Trans. Reliability*, vol. R-26, no. 4, October 1977, pp. 248-249.

[10] H. Nakazawa, "A decomposition method for computing system reliability by a Boolean expression," *IEEE Trans. Reliability*, vol. R-26, no. 4, October 1977, pp. 250-252.

[11] K. Aggarwal, J. Gupta and K. Misra, "A simple method for reliability evaluation of a communication system," *IEEE Trans. Communications*, vol. COM-23, No. 5, May 1975, pp. 563-566.

[12] K. Aggarwal, K. Misra and J. Gupta, "A fast algorithm for reliability evaluation," *IEEE Trans. Reliability*, vol. R-24, April 19675, pp. 83-85.

[13] J. Abraham, "An improved algorithm for network reliability," *IEEE Trans. Reliability*, vol. R-28, no. 1, April 1979, pp. 58-61.

[14] A. Satyanarayana and A. Prabhakar, "New topological formula and rapid algorithm for reliability analysis of complex networks," *IEEE Trans. Reliability*, vol. R-27, no. 2, June 1978, pp. 82-100.

[15] R. Miller, *Switching Theory, Volume 1: Combinational Circuits*, New York, Wiley, 1965.