A SYSTEM RELIABILITY INTERPRETIVE LANGUAGE

A Thesis

Presented to

the Faculty of the Department of Industrial Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Industrial Engineering

---

by

John J. Pothanikat

January 1970

523061

ACKNOWLEDGEMENTS

The author wishes to express his sincere gratitude to his advisor, Dr. Charles E. Donaghey, for his advice and guidance.

Most deeply, the author is indebted to his wife, Mrs. Tresa John for her assistance and continuing encouragement.

Thanks are extended to Mrs. Van Winkle for her patient efforts in the typing of this paper.

A SYSTEM RELIABILITY INTERPRETIVE LANGUAGE

An Abstract of a Thesis

Presented to

the Faculty of the Department of Industrial Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

John J. Pothanikat

January 1970

ABSTRACT

This interpretive language is designed to calculate the overall system reliability of large system configurations. There are no other such user oriented programs available to calculate system reliability. It is felt that this language may be utilized by systems designers in optimizing the overall reliability of systems with which he is concerned. With the commands such as Delete, Add, Modify, the designer can easily mainpulate his system in order to get the most economical and reliable system.

Any system configurations with units in pure series, pure parallel or a combination of these two can be handled with this interpretive language. This interpreter is written entirely in Fortran IV which allows implementation on almost any computer system.

TABLE OF CONTENTS

Chapter Page

CHAPTER I

INTRODUCTION

The problem of assuring reliable electrical circuit performance faces many designers in various industries. Circuit performance is especially critical in the aerospace industries, consumer electronics, and communications systems. The problem is one of attaining a desired reliability that is a compromise between economy and the minimum product reliability that the consumer will accept. In order to solve the above problem the system designer has to calculate the system reliability for all possible alternatives and combinations. At present there are no general user oriented computer programs available to handle the above mentioned problems efficiently. Each of the system reliability problems must be individually programmed and solved; however, this is both tedious and time consuming. The programming becomes more and more tedious and time consuming as the system becomes larger and larger. Hence it is felt there is a need for a general user's program which can alleviate these problems. SYRIL (SYSTEM RELIABILITY INTERPRETIVE LANGUAGE) has been developed to solve these problems. SYRIL allows a user with little background in programming to quickly and efficiently solve fairly large system reliability problems.

SYRIL is a very general and open-ended language. New commands can easily be added to the language. The present version

described in this paper can handle systems involving as many as 100 components. The present version of SYRIL assumes the components failure distributions are all independent.

The present version of SYRIL works on the principle of a reduction technique. This interpreter handles systems with units in pure parallel or pure series or a combination of these two. The following sketches illustrate the three different types of system configurations which are handled by SYRIL:

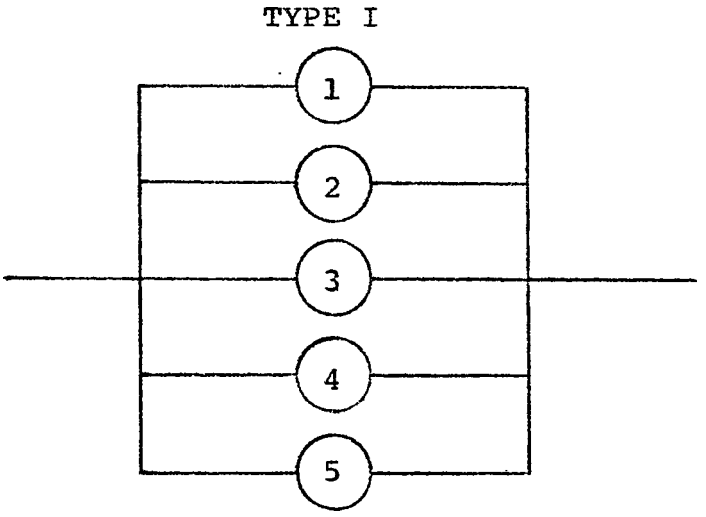

Figure (1) PURE PARALLEL SYSTEM

SYRIL reduces the above pure parallel system to a single equivalent unit by the following calculation

$$\text{System Reliability} = 1-(1-R_1)(1-R_2)(1-R_3)(1-R_4)(1-R_5)$$

where R_1, R_2, R_3, R_4, and R_5 are the reliabilities of the units 1, 2, 3, 4, and 5 respectively.

TYPE 2

Figure (2) PURE SERIES SYSTEM

Just as in the previous case SYRIL reduces the above pure series configuration to a single equivalent unit by the following formula

$$\text{System Reliability} = R_1 \cdot R_2 \cdot R_3 \cdot R_4$$

where R_1, R_2, R_3, and R_4 are reliabilities of the individual units 1, 2, 3, and 4 respectively.

TYPE 3

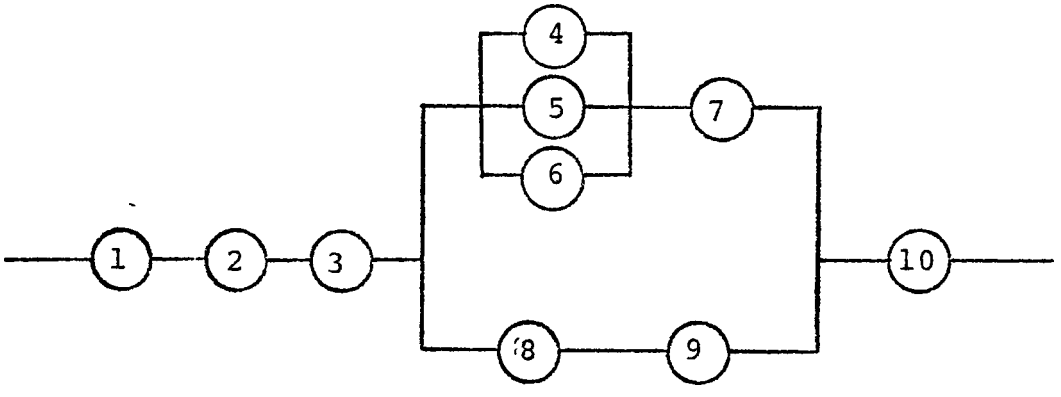

Figure (3)

SYSTEMS WITH UNITS IN A SERIES PARALLEL-COMBINATION

SYRIL, first of all, reduces the pure parallel units if any exists in the system. In the Fig. (3) the units 4, 5, and 6 are in pure parallel; hence, these three units are reduced to a single equivalent unit. Now the system will be as represented by the Fig. (4). The single equivalent unit is represented by R.

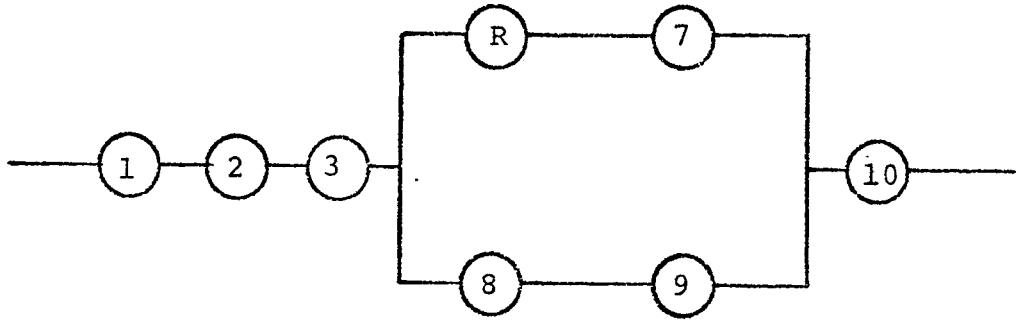

Figure (4)

SYRIL now reduces all the pure series units in the system. There are three different sets of pure series connected units namely 1, 2, and 3, units R and 7, and units 8 and 9. SYRIL reduces each of these three sets and forms up the equivalent single unit for each set by applying the series reduction formula. After the series reduction the system will look like Fig. (5).

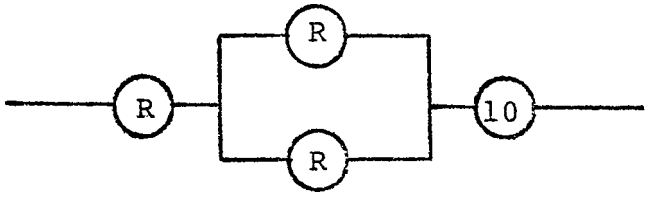

Figure (5)

SYRIL again searches for any units in pure parallel connection. There are two units connected in pure parallel and they are reduced to a single unit by the parallel reduction formula and the reduced system configuration will look like Fig. (6).

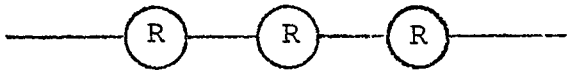

Figure (6)

Now the system contains only three units in pure series connection. SYRIL reduces these units to a single unit which represents the overall system reliability.

SYRIL applies the above reduction techniques to any larger systems just in the same way described above.

The following sketches are some of the system configurations which do not fall under the above catagory and thereby cannot be solved by this interpreter.

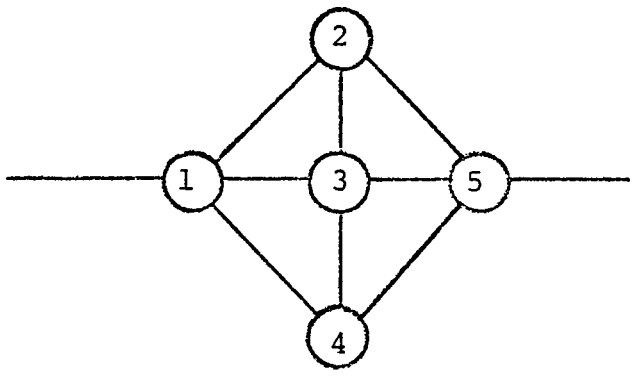

Figure (7)

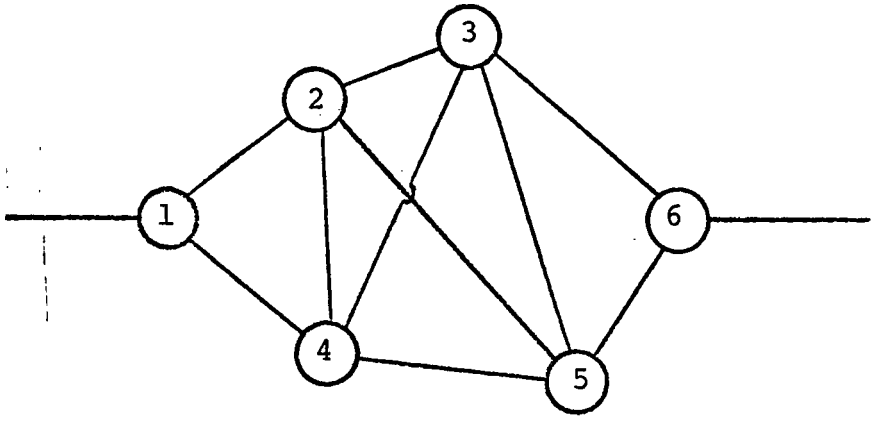

Figure (8)

However, SYRIL can be further developed to take care of such net works also. Signal flow reduction techniques may be a possible tool for simplifying the above complex networks to simple reducible ones.

In the present version, the user can furnish his system parametersin two different forms.

(1) The reliabilities of each of the units in the system can be funished, or

(2) The failure density function of each of the units and the operational time of the system can be furnished.

In the second case the user can calcualte the system reliability at any time of operation. Units with the following failure distributions can be presently worked out by SYRIL

(1) Weibull

(2) Exponential

(3) Normal

(4) Uniform

(5) Any distribution furnished by the user

(user has to feed in points from the CDF of the particular distribution)

Units with distributions other than what are shown above, can be worked out by modifying the interpreter appropriately.

This language is practically format free. The commands can be started in any column of the punch card, and any card can contain any number of commands. Each command must be terminated by a semi-colon. Blank spaces are ignored by the interpreter. The program is terminated by a single asterisk. Any number of problems can be fed in a single program; however each of the problems must be terminated by an asterisk.

The next section provides a description of the various rules regarding the input representation of the system configuration. New commands can be added to this interpreter as and when needed. SYRIL will solve reliabilities to four place accuracy.

CHAPTER II

The following rules are developed for the input representation of the system configuration.

Rule 1: Each of the units in the system is numbered as 1, 2, 3.....N. There are no precedence relations to be followed in this numbering. The system shown in Fig. (9) is numbered according to this rule.

Rule 2: Each unit is preceded by one number and followed by another different number. For example in the Fig. (9), unit 6 is preceded by 3 and followed by 4.

Rule 3: There should be only one number in between any two units.

Figure (9) is numbered according to these rules.

The following are the various commands currently available in the SYRIL language.

Section 1 of this chapter deals with all the commands for the case where the unit reliability of each of the units in the system is known. Where as section II deals with all the commands for the case where the unit failure distribution and its parameters are known. Each system reliability problem must be wholly described by the commands in section 1 or section 2.

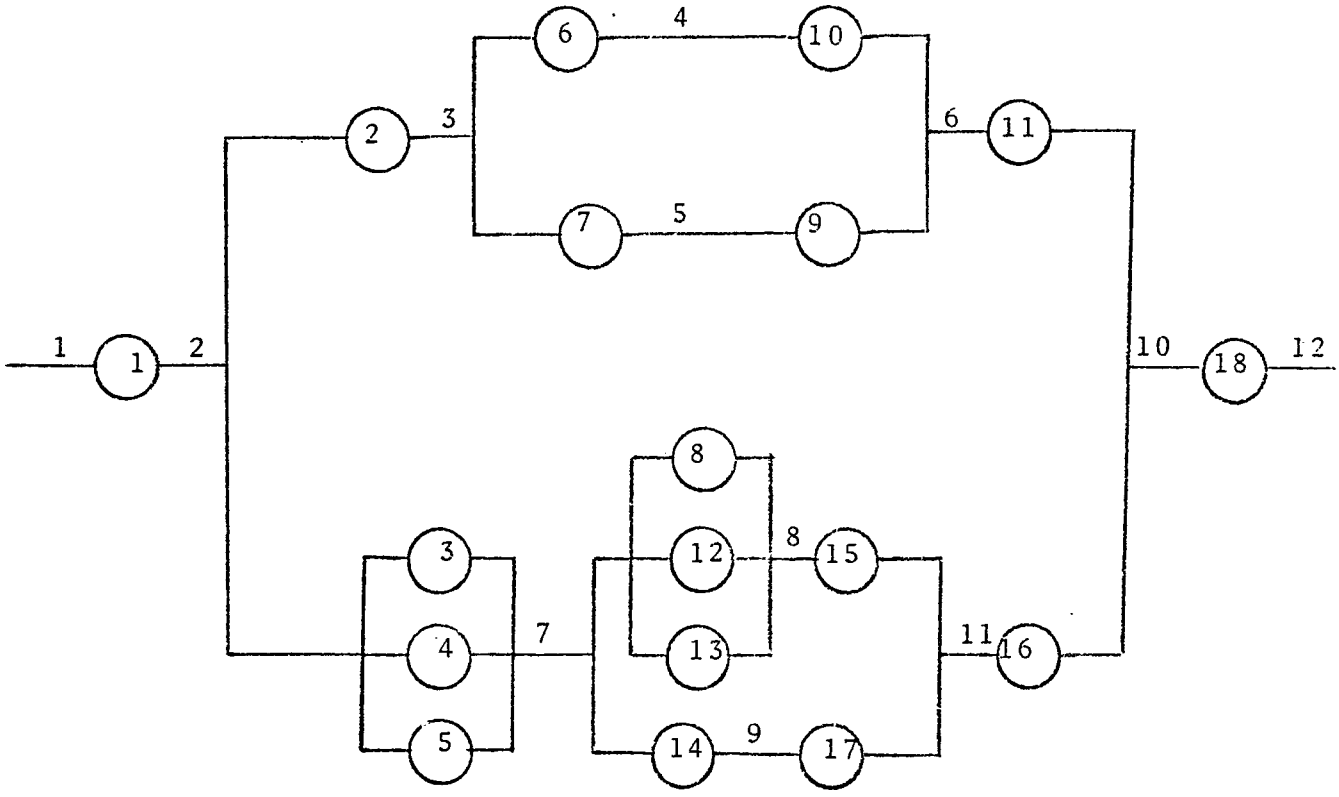

Figure (9)

Section 1

Input Commands - Each of the units in the system is to be described to the computer in the following form.

R<n where n represents the unit number according to the system numbering described previously >=< Reliability of the nth unit >,< N where N is the number which immediately precedes the nth unit >-< M where M is the number which immediately follows the nth unit >.

For example according to Figure (9) the input commands for the units 1, 2 and 3 would be as follows.

R1 = 0.9, 1-2;
R2 = 0.8, 2-3;
R3 = 0.8, 2-7;

where 0.9, 0.8 and 0.8 are the reliabilities of the units 1, 2 and 3 respectively.

Output Commands

SYSTEM RELIABILITY - In order to find the system overall reliability, the required command is <FIND SYSTEM RELIABILITY>;

MODIFY - In order to modify any of the unit reliabilities currently in the system, the command is <MODIFY> <Rn where n is the unit to be modified >= <The modified reliability of the nth unit >;

For example if the unit reliability of the 3rd unit is to be modified to 0.9, the corresponding command will be as follows.

MODIFY R3 = 0.9;

DELETE - To delete any of the units currently in the system the command is <DELETE> <Rn where n is the unit to be deleted>;

For example if the unit 6 is to be deleted, the command is.

DELETE R6;

ADD - To add a new unit to the current system the command is as follows. <ADD><Rn where n is the number of the new unit >=< Reliability of the unit >,<N where N is the number which immediately precedes the nth unit >-<M where M is the number which immediately follows the nth unit >;

For example according to Figure (9) if unit 19 is added to the system to the right of unit 18, the command will be as follows

ADD R19 = 0.8, 12-13;

R OUT OF N STRUCTURE - The following command can be used to calculate the system reliability for the case where at least R out of N units should be good. R should be less than or equal to N where N is the total number of units in the system. Each of the units must be identical, independent and in parallel.

The required command is <FIND SYSTEM RELIABILITY>(<R where R is the required minimum number of good units >/<N where N is the total number of the system >);

For example if the system contains 5 identical parallel units and if the system functions if and only if at least 2 out of 5 units are good. The system reliability for this particular case is found by using the following command.

FIND SYSTEM RELIABILITY (2/5);

LABEL - To label and store any system configurations the following command is to be used. Two system configurations can be stored at the same time under the label names SC1 and SC2.

<LABEL> SC n where n can be either 1 or 2.

To label and store the system overall reliability of any system the command is

<LABEL> SR n where n can be either 1 or 2.

At any time the overall system reliability of two systems can be stored under the label names SR1 and SR2.

NOTE: All the LABEL commands should be preceded by a FIND SYSTEM RELIABILITY command.

PRINT - In order to print out the labeled configuration the command is. <PRINT> SC n where n can be either 1 or 2 depending upon under which one that system was labeled.

For example if we want to get the print out of a system configuration labeled SC1, the command will be as follows.

PRINT SC1;

Similarly the system overall reliability can be printed out by using the following command

PRINT SR n where n can be 1 or 2 depending upon the label command;

TRANSFER - In order to transfer the labeled system configuration to the working storage locations the following command is to be used. <TRANSFER> SC n where n can be either 1 or 2 depending upon the label command.

This enables the user to get the stored configuration to the working cells and to manipulate the system for the further changes.

The user can transfer the labeled overall system reliability to any unit he wants by using the following command. <TRANSFER> <SR n where n can be either 1 or 2> TO <RL where L represents the unit number to which the system overall reliability is to be transferred> <NI where I is the number which immediately precedes the unit>, <MJ where J is the number which immediately follows the unit>;

By the application of the above command the user can treat the whole system as a single unit in another system.

Section 2

Input Commands - The following failure distributions are available to the SYRIL language.

1) Exponential
2) Normal. p.d.f. = $f(t) = \frac{1}{\sqrt{2\pi}\sigma}\left(\exp -\frac{1}{2}\left[\frac{t-\mu}{\sigma}\right]^2\right)$ $-\infty<t<\infty$
   where μ = mean and σ = standard deviation
3) Weibull. p.d.f. = $f(t) = Kt^m e^{-Kt^{m+1}/(m+1)}$
   where K = scale - change parameter and m = shape parameter.
4) Uniform
5) Any distribution provided by the user. In this case the user has to furnish points (maximum of 8) from the C.D.F.

The Input Command for the unit with exponential distribution is.

<R_n where n is the unit number> = <EXP>(<mean of the distribution which can be an integer or a real number>)<,><N where N is the number which immediately precedes the nth unit> - <M where M is the number which immediately follows the nth unit>;.

For example according to figure 9, if unit 2 is exponentially distributed with mean equal to 2.5, the input command is.

R2 = EXP(2.5), 2-3

The Input Command for Normal distribution is.

<Rn where n is the unit number> = <NOR>(<MEAN of the distribution>,<Standard deviation>)<,><N where N is the number which immediately precedes the nth unit> - <M where M is the number which immediately follows the nth unit>;

For example if the unit 3 in figure 9 is normally distributed with a mean of 4 and standard deviation of 1.5, the input command is.

R3 = NOR(4, 1.5), 2-7;

The Input Command for Weibul distribution is.

<Rn where n is the unit number> = <WEI>(<K where K is the scale parameter >, <m where m is the shape parameter >)<,><N where N is the number which immediately precedes the nth unit> - <M where M is the number which immediately follows the nth unit>;

Both k and m can be integer or real numbers.

The following is an example of an input command for a Weibul distribution unit.

R4 = WEI(1.5, 0.2), 2-7;

The Input Command for Uniform distribution is.

<Rn> = <UNI>(<lower value of the distribution>,<upper value of the distribution>)<,><N> - <M>;

For example, according to figure 9, if the unit 5 is uniformly distributed with lower and upper limits 5 and 100.5, the input command is.

R5 = UNI(5, 100.5), 2-7;

The Input Command for Cumulative distribution function is.

<Rn><CDF>(<T1, T2, ...Tn, R1, R2, ... Rn

where T1, T2, ...Tn are the time values corresponding to the points from a CDF and R1, R2, ... Rn are the cumulative probability of failure corresponding to the time values T1, T2, ...Tn. The maximum value of n can be 8>)<,><N> - <M>;

For example, if unit 9 of figure 9 has a cumulative failure distribution as shown by figure 10, the input command for unit 9 is as follows.

R9 = CDF(0.0, 10.0, 20.0, 30.0, 40.0, 0.0, 0.1, 0.2 0.3, 1.0), 5-6;

Output Commands

System Reliability - The system overall reliability for any operating time can be calculated by using the following command. <FIND SYSTEM RELIABILITY>(<T where T is the operation time which can be an integer or a real number><)>;

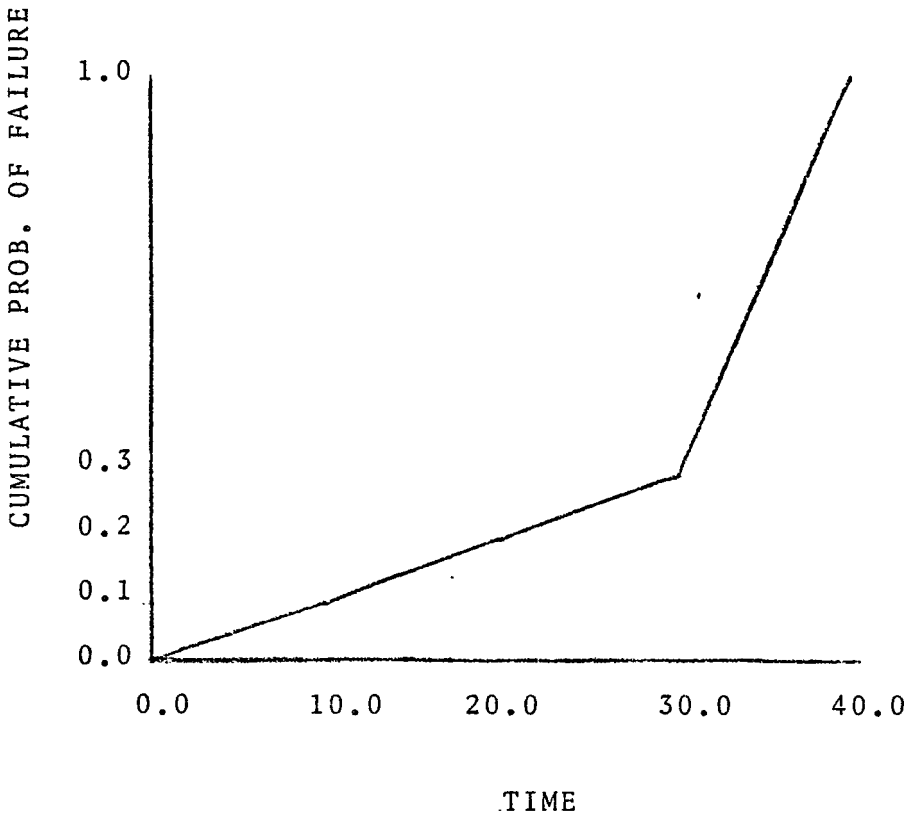

Figure (10)

For example, to find the system reliability for an operating time of 10 time units the command is.

FIND SYSTEM RELIABILITY (10);

MODIFY - In order to modify any of the failure distributions or parameters of the units currently in the system the command is <MODIFY> <Rn where n is the unit to be modified> = <The modified failure distribution>(<The parameters currently applicable according to the current distribution><)>.

For example, if unit 3 of figure 9 is to be modified to Exponential failure distribution with a mean of 2, the

required command is.

MODIFY R3 = EXP(2);

DELETE - To delete any of the units currently in the system the command is <DELETE> <Rn where n is the unit to be deleted>;

For example, if the 4th unit is to be deleted, the command is

DELETE R4;

ADD - To add a new unit to the current system, the command is as follows. <ADD> <Rn where n is the number of the new unit> = < failure distribution>(<parameters of the distribution>)<,><N where N is the number which immediately precedes the nth unit> - <M where M is the number which immediately follows the nth unit>;

For example, according to figure 9, if unit 19 is added to the system to the right of unit 18, the command is as follows.

ADD R19 = EXP(2.5), 12-13;

where the new unit is exponentially distributed with a mean of 2.5.

STANDBY - If each of the units in a parallel system has an identical exponential distribution life, then the standby reliability of the system can be found out by using the following command. Standby reliability infers that as each component fails, the one below it on the parallel structure begins its operation. <FIND STANDBY RELIABILITY>(<T where T is the operation time><)>; T can be an integer or a real number.

For example, the command

FIND STANDBY RELIABILITY (100.5);

finds out the standby system reliability for 100.5 time units of operation.

PLOT - To get a plot time vs. system reliability, the following command is to be used. <PLOT>(<Ts where Ts is the start time>, <TI where TI is the increments of time>,<TE where TE is the end of time>; all these time values can either be as integers or as real numbers.

For example, the command

PLOT (0.0, 5.5, 150);

will plot the graph time vs. system reliability with 0.0 as starting time, 5.5 as the increment in time and 150 as the end of the operation time.

COMMENTS - Comments can be inserted anywhere in a SYRIL program. They must, however, be followed by a semi-colon.

The next chapter deals with program description of the SYRIL interpreter.

The following are the computer outputs of a few SYRIL problems.

S Y R I L

EXAMPLE 1

```
THIS IS AN EXAMPLE OF SYSTEM RELIABILITY CALCULATIONS
INDIVIDUAL UNIT RELIABILITIES ARE GIVEN;

R1=0.9,1-2;
R2=0.8,2-3;
R3=0.7,2-7;
R4=0.8,2-7;
R5=0.7,2-7;
R6=0.8,3-4; R7=0.9,4-5;
R10=0.8,5-11;
R8=0.9,3-6; R9=0.8,6-5;
R11=0.9,7-8; R12=0.9,7-8;
R13=0.9,8-11;
FIND SYSTEM RELIABILITY;
LABEL SC1; LABEL SR1;
MODIFY R4=0.9;
FIND SYSTEM RELIABILITY;
DELETE R3;
FIND SYSTEM RELIABILITY;
ADD R14=0.8,11-12;FIND SYSTEM RELIABILITY*
```

SYSTEM CONFIGURATION
****

UNIT NO.	R(I)	N(I)	M(I)
1	0.9000	1	2
2	0.8000	2	3
3	0.7000	2	7
4	0.8000	2	7
5	0.7000	2	7
6	0.8000	3	4
7	0.9000	4	5
8	0.9000	3	6
9	0.8000	6	5
10	0.8000	5	11
11	0.9000	7	8
12	0.9000	7	8
13	0.9000	8	11

****

SYSTEM RELIABILITY=0.8538 N= 1 M= 11

****

SYSTEM CONFIGURATION
****

UNIT NO.	R(I)	N(I)	M(I)
1	0.9000	1	2
2	0.8000	2	3
3	0.7000	2	7
4	0.9000	2	7
5	0.7000	2	7
6	0.8000	3	4
7	0.9000	4	5
8	0.9000	3	6
9	0.8000	6	5
10	0.8000	5	11
11	0.9000	7	8
12	0.9000	7	8
13	0.9000	8	11

****

SYSTEM RELIABILITY=0.8568 N= 1 M= 11

****

SYSTEM CONFIGURATION
****

UNIT NO.	R(I)	N(I)	M(I)
1	0.9000	1	2
2	0.8000	2	3
3	0.0	2	7
4	0.9000	2	7
5	0.7000	2	7
6	0.8000	3	4
7	0.9000	4	5
8	0.9000	3	6
9	0.8000	6	5
10	0.8000	5	11
11	0.9000	7	8
12	0.9000	7	8
13	0.9000	8	11

****

SYSTEM RELIABILITY=0.8499 N= 1 M= 11

****

SYSTEM CONFIGURATION
****

UNIT NO.	R(I)	N(I)	M(I)
1	0.9000	1	2
2	0.8000	2	3
3	0.0	2	7
4	0.9000	2	7
5	0.7000	2	7
6	0.8000	3	4
7	0.9000	4	5
8	0.9000	3	6
9	0.8000	6	5
10	0.8000	5	11
11	0.9000	7	8
12	0.9000	7	8
13	0.9000	8	11
14	0.8000	11	12

****

SYSTEM RELIABILITY=0.6799 N= 1 M= 12

****

```
      EXAMPLE 2    ;
 THE FOLLOWING  PROBLEM IS AN EXAMPLE  OF SYSTEM RELIABILITY CALCULATION
 FOR AN R OUT OF N SYSTEM;


R1=0.8,1-2;
R2=0.8,1-2;
R3=0.8,1-2;
R4=0.8,1-2;
R5=0.8,1-2;
FIND SYSTEM RELIABILITY(3/5);
FIND SYSTEM RELIABILITY*
```

```
**************************************************

SYSTEM RELIABILITY FOR    3 OUT OF  5 UNITS =0.9421

**************************************************
```

SYSTEM CONFIGURATION
****

UNIT NO.	R(I)	N(I)	M(I)
1	0.8000	1	2
2	0.8000	1	2
3	0.8000	1	2
4	0.8000	1	2
5	0.8000	1	2

****

SYSTEM RELIABILITY=0.9997 N= 1 M= 2

****

```
      EXAMPLE 3;
PRINT SR1; PRINT SC1;
TRANSFER SC1;FIND SYSTEM RELIABILITY;
TRANSFER SR1 TO R2,N2,M3;
R1=0.8,1-2; R3=0.9,3-4;
FIND SYSTEM RELIABILITY*
```

**** SYSTEM OVER ALL RELIABILITY BETWEEN NODES 1 AND 11 = 0.8538 ****

SYSTEM CONFIGURATION
****

UNIT NO.	R(I)	N(I)	M(I)
1	0.9000	1	2
2	0.8000	2	3
3	0.7000	2	7
4	0.8000	2	7
5	0.7000	2	7
6	0.8000	3	4
7	0.9000	4	5
8	0.9000	3	6
9	0.8000	6	5
10	0.8000	5	11
11	0.9000	7	8
12	0.9000	7	8
13	0.9000	8	11

SYSTEM CONFIGURATION
****

UNIT NO.	R(I)	N(I)	M(I)
1	0.9000	1	2
2	0.8000	2	3
3	0.7000	2	7
4	0.8000	2	7
5	0.7000	2	7
6	0.8000	3	4
7	0.9000	4	5
8	0.9000	3	6
9	0.8000	6	5
10	0.8000	5	11
11	0.9000	7	8
12	0.9000	7	8
13	0.9000	8	11

****

SYSTEM RELIABILITY=0.8538 N= 1 M= 11

****

SYSTEM CONFIGURATION
****

UNIT NO.	R(I)	N(I)	M(I)
1	0.8000	1	2
2	0.8538	2	3
3	0.9000	3	4

****

SYSTEM RELIABILITY=0.6148 N= 1 M= 4

****

EXAMPLE 4;

```
R1=NOR(185,5),1-2;
R2=EXP(210),2-3;
R3=UNI(0,280),3-4;
R4=EXP(198),4-5;
R5=CDF(0.0,90.0,152.0,230.0,300.0,0.0,0.2,0.4,0.6,1.0),3-5;
R6=NOR(250,7),3-6;
R7=WEI(0.02,0.01),6-5;
FIND SYSTEM RELIABILITY(8.5);
ADDR8=EXP(258),5-7;
FIND SYSTEM RELIABILITY(10.3);
MODIFY R6=EXP(278);
FIND SYSTEM RELIABILITY(10.3);
PLOT(0.0,7,100)*
```

SYSTEM CONFIGURATION
****

UNIT NO.	R(I)	N(I)	M(I)
1	1.0000	1	2
2	0.9603	2	3
3	0.9696	3	4
4	0.9580	4	5
5	0.9811	3	5
6	1.0000	3	6
7	0.8099	6	5

****

SYSTEM RELIABILITY FOR 8.5 TIME UNITS
OF OPERATION=0.9601 N= 1 M= 5

****

SYSTEM CONFIGURATION
****

UNIT NO.	R(I)	N(I)	M(I)
1	1.0000	1	2
2	0.9521	2	3
3	0.9632	3	4
4	0.9493	4	5
5	0.9771	3	5
6	1.0000	3	6
7	0.7784	6	5
8	0.9609	5	7

****

SYSTEM RELIABILITY FOR 10.3 TIME UNITS

OF OPERATION=0.9145 N= 1 M= 7

****

SYSTEM CONFIGURATION
****

UNIT NO.	R(I)	N(I)	M(I)
1	1.0000	1	2
2	0.9521	2	3
3	0.9632	3	4
4	0.9493	4	5
5	0.9771	3	5
6	0.9636	3	6
7	0.7784	6	5
8	0.9609	5	7

****

SYSTEM RELIABILITY FOR 10.3 TIME UNITS
OF OPERATION=0.9144 N= 1 M= 7

****

OPERATION TIME	SYSTEM RELIABILITY
0.0	1.0000
7.00	0.9412
14.00	0.8851
21.00	0.8313
28.00	0.7795
35.00	0.7298
42.00	0.6821
49.00	0.6364
56.00	0.5929
63.00	0.5515
70.00	0.5122
77.00	0.4752
84.00	0.4402
91.00	0.4071
98.00	0.3746

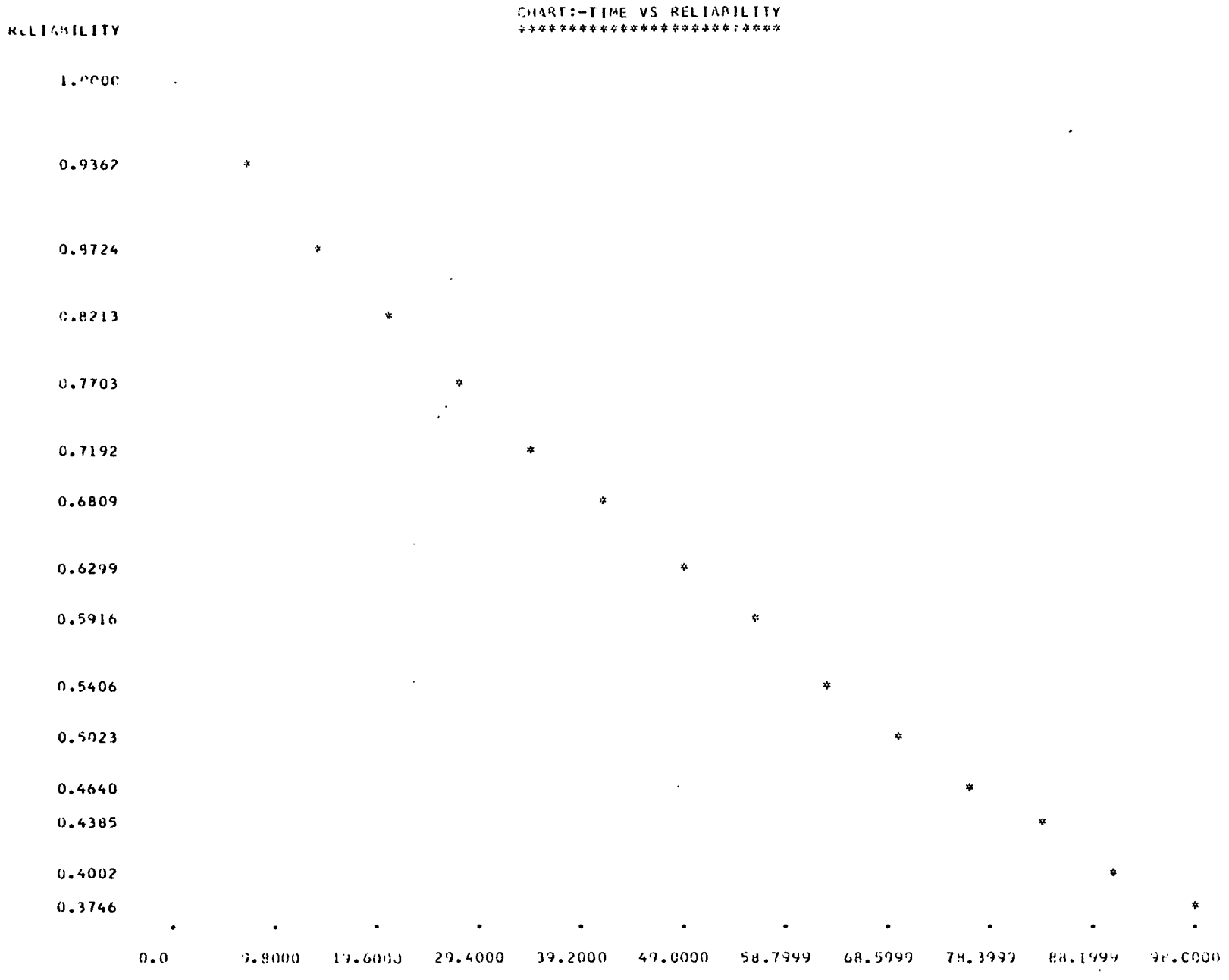
CHART:-TIME VS RELIABILITY
RELIABILITY
1.0000
0.9362
0.8724
0.8213
0.7703
0.7192
0.6809
0.6299
0.5916
0.5406
0.5023
0.4640
0.4385
0.4002
0.3746
0.0
9.8000
19.6000
29.4000
39.2000
49.0000
58.7999
68.5999
78.3999
88.1999
98.0000
UNITS OF TIME -------->

```
      EXAMPLE 5;

R1=EXP(150),1-2;
R2=EXP(150),1-2;
R3=EXP(150),1-2;
R4=EXP(150),1-2;
R5=EXP(150),1-2;
FIND STAND BY RELIABILITY(180)*
```

```
**********************************************************

STANDBY SYSTEM RELIABILITY FOR    180.0 TIME UNITS

OF OPERATION=0.9923

**********************************************************
```

CHAPTER III

Figure II represents the flow chart of the SYRIL interpreter. The first operation executed is initialization. A number of registers and flags are initially set, and a heading printed for the listing that is to follow. Subroutine INPUT is then called. INPUT reads the first card under a FORTRAN (80 A1) format, and the contents of the card are immediately printed out in the same manner. The characteristics are then placed one by one into a vector named K, capable of holding 2,000 elements. Any blank that is encountered is omitted from the vector. INPUT continues the above operation until it encounters an asterisk. Whenever an asterisk is encountered, a pointer is set to the first position in K. Subroutine INPUT then tests the current problem and finds out if it is a type I or a type II problem. That is if the unit reliability is given or if the failure distributions and parameters are given. If it is the first case subroutine, INPUT calls subroutine UINTER, otherwise subroutine FINTER is called.

When subroutine UINTER is called, a pointer has been set to the beginning of a SYRIL command in the K vector. UINTER examines the command and calls the appropriate subroutine that will execute the command. If the command is not found in the repertoire, subroutine UBAD is called, which merely drives the

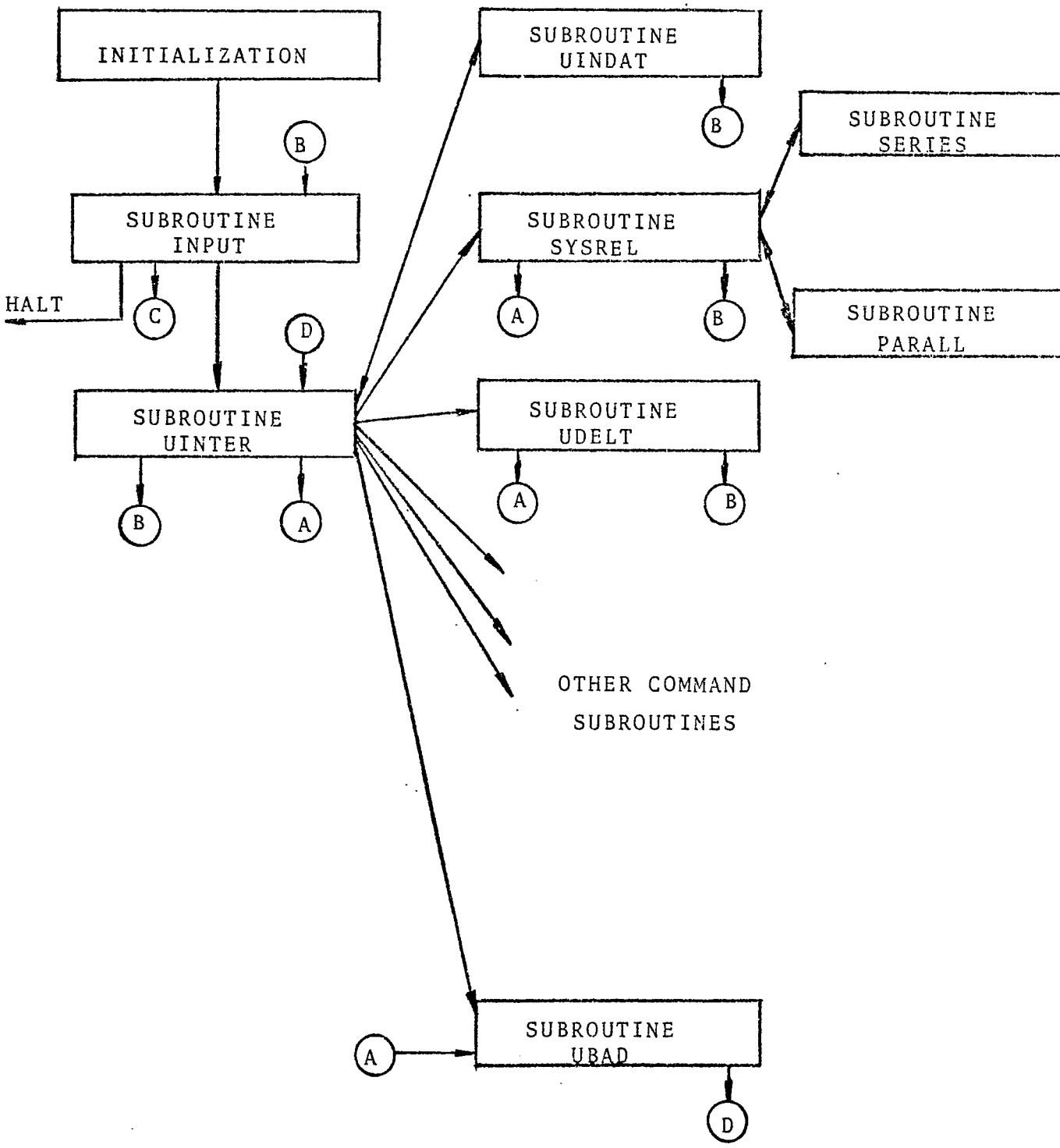

Figure (11) FLOW CHART OF SYRIL INTERPRETER

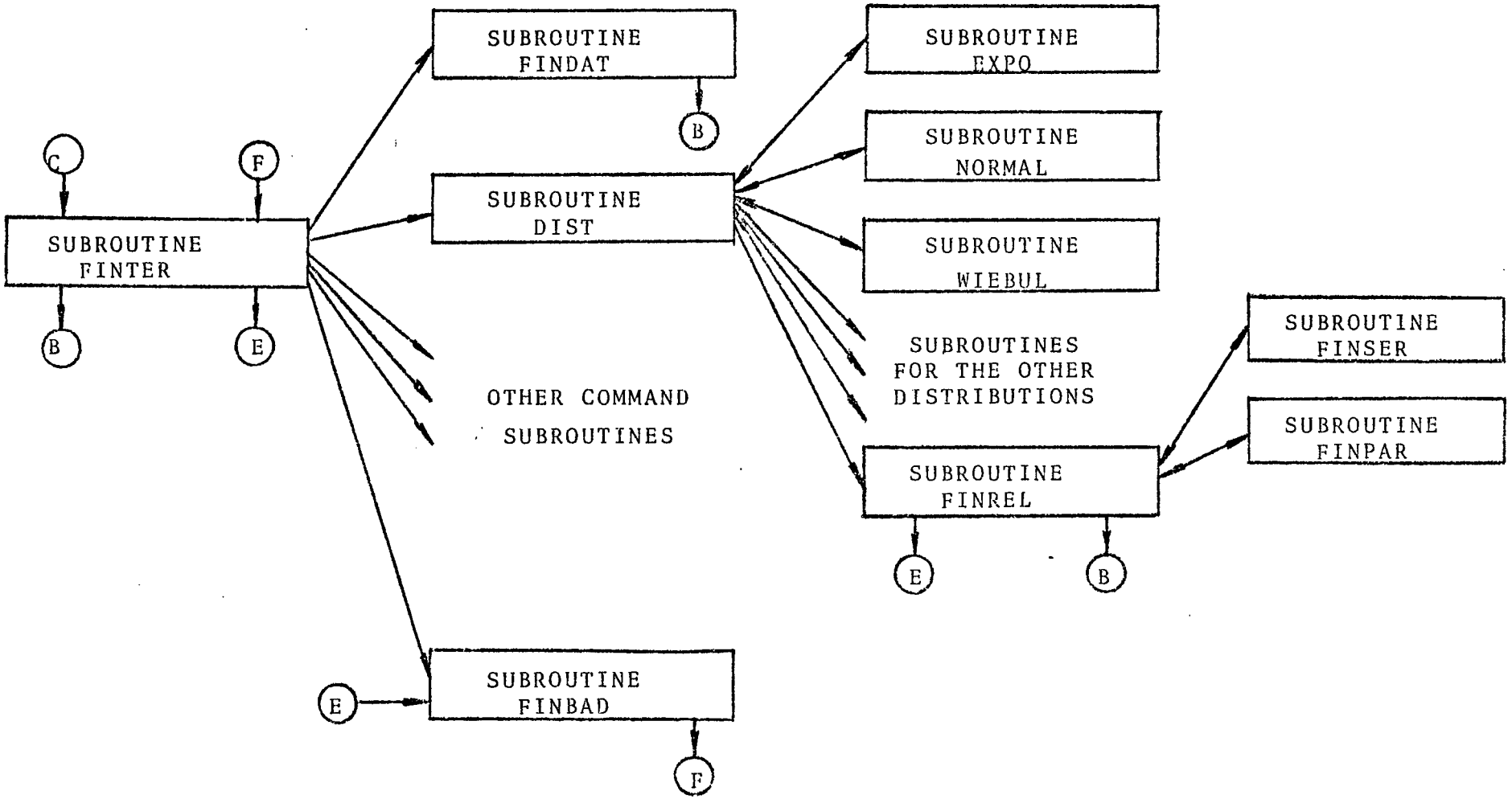

Figure (11) (Cont'd)

pointer to the next semi-colon in the K vector, and then calls subroutine UINTER. This is how comments are handled in SYRIL. The comment is not recognized as a legitimate SYRIL command by UINTER and hence control is transferred to subroutine UBAD which merely moves the pointer to the end of the comment.

The various command subroutines perform the operations on the arrays specified in the command. For example, the command R1 = 0.8, 1-2; will cause subroutine UINDAT to be called, and UINDAT stores the unit reliability, the number which precedes the unit and the number which follows the unit, in the appropriate storage cells.

The command MODIFY R5 = 0.8 will cause subroutine UMODIF to be called, and UMODIF changes the unit reliability of the 5th unit to 0.8.

Similarly, the command FIND SYSTEM RELIABILITY will cause subroutine SYSREL to be called. SYSREL initializes the necessary arrays and pointers and calls subroutine SERIES. Subroutine SERIES reduces all the pure series units in the system and returns the reduced system to subroutine SYSREL. SYSREL now calls subroutine PARALL. PARALL reduces all the pure parallel units in the system and returns control to subroutine SYSREL. SYSREL repeats the above operations until the whole system is reduced to a single equivalent unit. SYSREL now prints out the whole system configuration and the system overall reliability. If the system is non-reduceable, due to some errors in the system configuration, an error message is printed out

and the control is transferred to the subroutine INPUT which thereby starts on the next problem if any. SYRIL has been equipped with error messages for each of the available commands.

Each problem is terminated any time UINTER or UBAD encounters an asterisk.

When subroutine FINTER is called, a pointer has been set to the beginning of a SYRIL command in the K vector. FINTER examines the command and calls the appropriate subroutine that will execute the command. Whenever the command is not found, subroutine FINBAD is called, which merely drives the pointer to the next semi-colon in the K vector, and then calls subroutine FINTER. The command R1 = EXP(2.2), 1-2; will cause subroutine FINDAT to be called which stores the distribution, parameters and the numbers which precede and follow unit 1. After executing the command, subroutine FINBAD is called, which derives the pointer to the next semi-colon in the K vector, and transfers control to subroutine FINTER.

The command FIND SYSTEM RELIABILITY (20); will cause subroutine DIST to be called, which sets the operation time equal to 20 time units. DIST then tests the distribution of each of the units in the system and calls the respective subroutines and finds the reliability of each of the units for 20 time units of operation. DIST then calls subroutine FINREL. FINREL initializes the necessary arrays and pointers

and then calls subroutine FINSER. FINSER reduces all the pure series units in the system and returns to subroutine FINREL. FINREL then calls subroutine FINPAR which reduces all the pure parallel units and returns to subroutine FINREL. Subroutine FINREL repeats the above operations until the whole system is reduced to a single equivalent unit. FINREL prints out the whole system configuration and the system overall reliability for the operation time.

Similarly, the command PLOT (0.0, 10, 100); will cause subroutine MATRIX to be called which first of all stores the starting value, increment and the end value of the operation time. MATRIX then calls the subroutine for each of the unit failure distributions and calculates the unit reliability for the initial operating time. MATRIX then calls subroutine FINREL which calculates the overall system reliability for the initial time value and returns the control to subroutine MATRIX. MATRIX similarly calculates the system overall reliabilities for each of the incremented time units and then prints out the table of values representing the operation time and system reliability. MATRIX then calls subroutine PLOT which plots time vs. System overall reliability. PLOT then transfers the control to subroutine FINBAD, which as usual, drives the pointer to the next semi-colon and calls subroutine FINTER. If any of the commands are not executable due to some errors in the command, an appropriate error message will be printed

and control will be transferred to subroutine INPUT which starts on a new problem.

Each problem is terminated any time FINTER or FINBAD encounters an asterisk.

The present version of SYRIL does assume the entire system is core resident and it requires about 15k memory cells. SYRIL has been entirely written in FORTRAN IV, and the program has been successfully compiled and utilized on both an XDS SIGMA 7 and an IBM 360/MOD 44 computer system.

CONCLUSION

The SYRIL language has been designed to calculate the overall system reliability of system configurations. Syril has been utilized successfully on both an XDS Sigma 7 computer system, and on an IBM 360/44 computer system.

It would be much more useful to use this language in a conversational mode. By this the user would get quick responses for each of his commands. The commands such as Add, Modify and Delete would be much more meaningful to the user under a conversational mode. The present time sharing system at the University of Houston does not allocate sufficient memory to the conversational programs so that SYRIL could be operated in this manner. However, SYRIL could be adapted to a conversational mode with very little change.

The label commands described in this thesis show that it is possible to store system configurations in the memory for future manipulations. Currently SYRIL can store two system configurations in memory at the same time. The labeled configurations are stored in core itself. By storing the labeled configurations on external devices such as tapes or discs, it could be possible to store many system configurations in the computer without requiring any additional memory.

The present version of SYRIL is quite useful in the educational environment. People without any computer background can solve large system networks with very limited training.

The following are few of the future developments which can be made onto this language.

The total number of units (in a system) which SYRIL can work out is 100. But this can be raised to any other convenient number very easily.

Presently SRIL can work on units with the following failure distributions.

1) Exponential
2) Weibul
3) Normal
4) Uniform and
5) C.D.F.

SYRIL has been developed as a very open-ended language and other failure distributions such as Gamma, Rayleigh, etc. can be added quite easily as the needs of the users are discovered.

The present version of SYRIL works only on systems with units in pure series, pure parallel and with units in a combination of these two. It is felt that it would have been an added advantage to have solved other types of system networks which do not fall under this category. A few of such system networks are illustrated in the introduction of this thesis. Signal flow reduction technique may be a possible tool to get

those system networks to reduceable ones. This would be a worthwhile field of research and investigation.

Simulation technique might be another possible field of expansion. It would be easier to solve a very complex system network by simulation rather than with other techniques.

BIBLIOGRAPHY

1. Barlow R., F. Proschan. Mathematical Theory of Reliability. New York: John Wiley and Sons, Inc., 1965.

2. Bazovsky, I. Reliability Theory and Practice. Engle Woods Cliffs, New Jersey: Prentice-Hall, Inc., 1961.

3. Donaghey, Charles E., Osman S. Ozkul. STIL Users Mannual. Project Themis Information Processing System, 1968.

4. Fortran IV-H Reference Manual for SDS Sigma 5/7 Computers, California, 1968.

5. Hillier, Frederick S., Gerald J. Lieberman. Introduction to Operations Research. San Francisco: Holden-Day, Inc., 1968.

6. Kuo, Shan S. Numerical Methods and Computers. Massachusetts: Addison-Wesley Publishing Company, 1965.

7. Mathematical Routine Technical Mannual for SDS Sigma 5/7 Computers, California, 1968.

8. Meyer, Paul L., Introduction to Probability and Statistical Applications. Massachusetts: Addison-Wesley Publishing Company, 1966.

9. Numerical Subroutine Package. Technical Manual for SDS Sigma 5/7 Computers. SDS Users Group, Scientific Data Systems, California, 1968.

10. SDS Sigma 7 Computer Reference Manual, Scientific Data Systems, Xerox Company, California, 1968.

11. Shooman, Martin L. Probabilistic Reliability, An Engineering Approach. New York: McGraw-Hill Series in Electrical and Electronic Engineering, 1968.

12. Wine, R. Lowell. Statistics for Scientists and Engineers. New Delhi: Prentice-Hall of India (private) Ltd., 1966.

APPENDIX A

ERROR MESSAGES

SYRIL has been equipped with an error message for each of the available commands. If any one of the commands does not conform to the rules described in Chapter 2, an error message, stating that the particular command is in error, will be printed out. If the system configuration is non-reducable, the following message will be printed out.

INPUT CONFIGURATION IN ERROR,

If the input command does not conform to the rules, the following message will be printed out.

INPUT COMMAND IN ERROR

There are error messages for each of the output commands. The following are the listings of the available error messages.

1) INPUT COMMAND IN ERROR
2) INPUT CONFIGURATION IN ERROR
3) DELETE COMMAND IN ERROR
4) MODIFY COMMAND IN ERROR
5) ADD COMMAND IN ERROR
6) LABEL COMMAND IN ERROR
7) PRINT COMMAND IN ERROR
8) TRANSFER COMMAND IN ERROR

9) FIND SYSTEM RELIABILITY COMMAND IN ERROR

10) FIND SYSTEM RELIABILITY (R/N) COMMAND IN ERROR

11) FIND STANDBY RELIABILITY COMMAND IN ERROR

12) PLOT COMMAND IN ERROR