NASA Technical Memorandum 87593

The SURE Reliability Analysis Program

Ricky W. Butler

FEBRUARY 1986

NASA

NASA Technical Memorandum 87593

The SURE Reliability Analysis Program

Ricky W. Butler
Langley Research Center
Hampton, Virginia

NASA
National Aeronautics
and Space Administration

Scientific and Technical
Information Branch

1986

CONTENTS

INTRODUCTION

A reliability analysis of a reconfigurable fault-tolerant computer system inevitably requires the determination of the death-state probabilities of a stochastic reliability model. For more than a decade, automated tools (e.g., ARIES, SURF, CARE III, etc.) have been developed to analyze such models. (See ref. 1.) Recently, a new mathematical theorem was proven that enables the efficient computation of the death-state probabilities of a large family of semi-Markov models that are useful for the reliability analysis of fault-tolerant architectures. (See ref. 2.) A major advantage of this new approach is that an arbitrary recovery transition can be handled. Consequently, a specific parametric form of the distribution, such as exponential or uniform, does not have to be assumed. This theorem served as the basis of the original version of the Semi-Markov Unreliability Range Evaluator (SURE). (See ref. 3.) After the development of the original SURE program, the mathematical technique was generalized by Lee (ref. 4) and White (ref. 5). The generalized method has been used to expand the capabilities of the SURE program.

Both White's and Lee's methods provide a means for bounding the probability of entering a death state of a semi-Markov model using simple parameters of the model such as the means and variances of the transitions. Consequently, the SURE program computes an upper and lower bound on system reliability. Although, an exact answer is not produced by the SURE program, the calculated bounds are close together for reliability models of ultrareliable systems - usually within 5 percent of each other. The advantage of the SURE technique is that the bounds are algebraic in form and, consequently, are computationally efficient. Very large and complex models can be analyzed by the program. Furthermore, the technique applies to the general class of semi-Markov models and thus does not impose restrictions on the type of architecture that can be analyzed. Of course, the practical utility of the tool is related to the closeness of the generated bounds.

Since SURE can handle a general distribution of recovery time, the overall fault-handling process of a fault-tolerant computer system can be captured in a single transition. It is unnecessary to assume some underlying parametric form or a special model of fault-handling behavior. The results of experimentation can be directly utilized. However, if desired, detailed fault-handling models can be incorporated into the system reliability model and analyzed by SURE.

In this paper, the essential details of White's and Lee's bounding techniques are presented along with a detailed description of the user interface to the SURE program.

Reliability Modeling of Computer System Architecture

Highly reliable systems must use parallel redundancy to achieve their fault tolerance since current manufacturing techniques cannot produce circuitry with adequate reliability. Furthermore, reconfiguration has been utilized in an attempt to increase the reliability of the system without the overhead of even more redundancy. Such systems exhibit behavior that involves both slow and fast processes. When these systems are modeled stochastically, some state transitions are many orders of magnitude faster than others. The slower transitions correspond to fault arrivals in the system. If the states of the system are delineated properly, then the slow transitions can be obtained from field data and/or by using the MIL-STD-217D Handbook calculation. These transitions are assumed to be distributed exponentially. The faster transition rates correspond to the system response to fault arrivals and can be measured experimentally using fault injection. (Experiments made by the Charles Stark Draper Laboratory, Inc., on the Fault-Tolerant Multiprocessor (FTMP), computer architecture have demonstrated (see ref. 6) that these transitions are not exponential.) Once a system has been mathematically modeled and the state transitions determined, a computational tool such as SURE may be used to compute the probability of entering the death states (i.e., the states that represent system failure) within a specified mission time, e.g., 10 hours.

Mathematical models of fault-tolerant systems must describe the processes that lead to system failure and the system fault-recovery capabilities. The first level of model granularity to consider is thus the unit of reconfiguration/redundancy in the system. In some systems this is as large as a complete processor with memory. In other systems, a smaller unit such as a CPU or memory module is appropriate. The states of the mathematical model are vectors of attributes such as the number of faulty units, the number of removed units, etc. Certain states in the system represent system failure and others represent fault-free behavior or correct operation in the presence of faults.

A semi-Markov model of a triad of processors with one spare is given in figure 1. The outputs of the processors in the triad are voted in order to mask faults. (In this model it is assumed that the spares do not fail while inactive.) The horizontal transitions represent fault arrivals. These occur with exponential rate λ. The coefficients of λ represent the number of processors in the configuration that can fail. The vertical transitions represent recovery from a fault. A recovery transition typically is not exponentially distributed and, consequently, its rate is time dependent: $\delta(t)$. (This time must be local time, i.e., the time since entering the current state, in order to preserve the semi-Markov property of the system.) Since the system uses three-way voting for fault masking, there is a "race" between

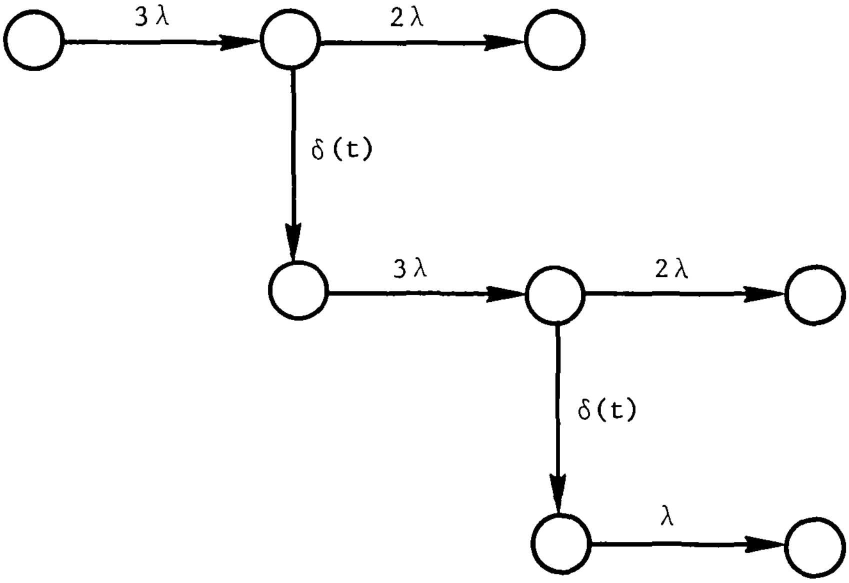


Figure 1.- Semi-Markov model of a triad with one spare.

the occurrence of a second fault and the removal of the first. If the second fault wins the race, then system failure occurs.

The SURE Program

Traditionally, the calculation of the probability of entering a death state of a semi-Markov model requires the solution of a set of coupled differential equations. Furthermore, because of the large disparity between the rates of fault arrivals and system recoveries, models of fault-tolerant architectures inevitably lead to numerically stiff differential equations. This problem along with the large computational cost of solving large state space problems have led to the use of exotic computational methods in recent reliability analysis tools such as CARE III and HARP. (See refs. 7 and 8.) In such programs, the problem is decomposed into a fault-handling model and a fault-occurrence model. Coverage parameters derived from the solution of the fault-handling model are inserted by various aggregation techniques into the fault-occurrence model in order to compute the system reliability. The coverage parameters are computed based on the assumption that critical-pair failures are the dominant failure mode in the system. Unfortunately, such strategies reduce the class of architectures that can be modeled. Because SURE does not rely on the solution of differential equations, stiffness is not a problem. In fact, the "stiffer" the model, the more accurate the approximation technique. Furthermore, since the SURE program computes by using algebraic formula, large state spaces can be accommodated. Therefore, decomposition or aggregation techniques are unnecessary and have not been utilized. A simple model pruning technique, however, has been included in the SURE program for extremely large models that otherwise might require large computational resources.

The first computational method developed by White formed the basis of the original SURE (version 1) program. (See ref. 3.) This program processed a simple input language that described a semi-Markov model and computed the death-state probabilities by using White's theory. The bounds were found to be very close for models of ultrareliable systems, i.e., those with slow exponential transitions (which describe fault occurrence) and general transitions that are fast with respect to the mission time (which describe recovery processes). The primary deficiencies of the first version of SURE were the inability to analyze models containing states with multiple competing recovery transitions and the inability to handle models with renewal processes (e.g., transient faults). By utilizing the generalized methods of White and Lee, both of these deficiencies have been eliminated in the current version of SURE. The following is a list of improvements of SURE (version 2) over the original version:

1. The program can now handle multiple recovery transitions from a state.

2. The program can now process models that are not pure death processes. Consequently, transient fault models can be analyzed.

3. Transitions can now be described by nonlinear functions of the variable as well as by linear functions.

4. Recovery distributions can be described by using either means and variances or means and percentiles.

5. User-controlled model truncation strategies are supported.

6. The computations are performed using double-precision arithmetic.

The SURE program is currently running under VMS 3.7 on VAX-11/750 and VAX-11/780 computers at the NASA Langley Research Center. The program has been designed with minimal usage of VMS specific constructs. Consequently, the program should be easy to transfer to other systems. The SURE program consists of three modules — the front end module, the computation module, and the graphics output module. The front end and computation modules are implemented in Pascal and should easily transfer to other machines. The graphics output module is written in FORTRAN but uses the graphics library TEMPLATE; this module can be used only by installations having this library. The SURE program can be installed and used without the graphics output module. Alternatively, this module can be rewritten using another graphics library.

BOUNDS BASED ON MEANS AND VARIANCES

In this section White's theorem (ref. 5) will be discussed but not proven. White's theorem involves a graphical analysis of a semi-Markov model. The theorem provides a means of bounding the probability of traversing a specific path in the model within the specified time. By applying the theorem to every path of the model, the probability of the system reaching any death state can usually be determined within very close bounds. A simple semi-Markov model of the six-processor Software Implemented Fault Tolerance (SIFT) computer system (see ref. 9) will be used to introduce the theorem. This model is illustrated in figure 2. The horizontal

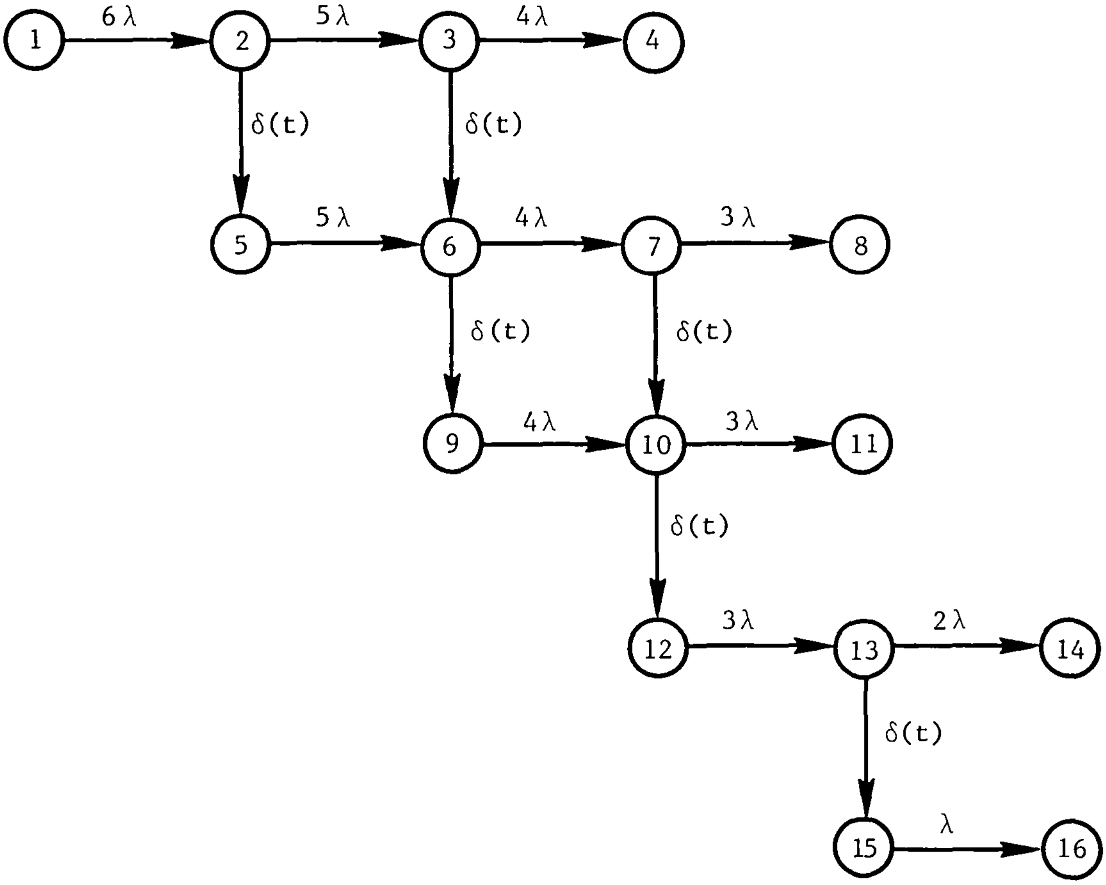


Figure 2.- Semi-Markov model of SIFT.

transitions in the model represent fault arrivals. These are assumed to be exponentially distributed and relatively slow. The vertical transitions represent system recoveries by reconfiguration, i.e., removal of the faulty processor from the working set of processors. These transitions are assumed to be fast but can have arbitrary distribution. White's theorem requires only that the mean and variance of the recovery time distribution and its transition probability be specified. The death states of the model are states 4, 8, 11, 14, and 16. Death state 4 represents the case in which three processors out of six have failed before the system reconfigures. State 16 represents the case in which the system has been completely depleted of processors. The unreliability of the system is precisely the sum of the probabilities of entering each death state. White's theorem analyzes every path to each death state individually. In the SIFT model the following paths must be considered:

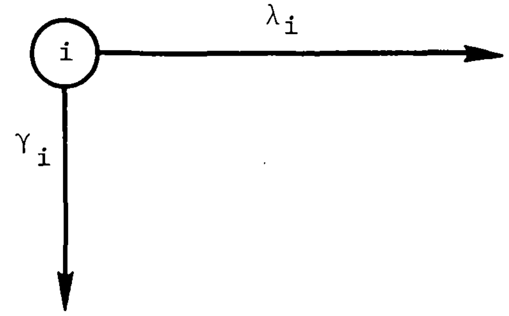
path 1:	1 -> 2 -> 3 -> 4
path 2:	1 -> 2 -> 3 -> 6 -> 7 -> 8
path 3:	1 -> 2 -> 5 -> 6 -> 7 -> 8
path 4:	1 -> 2 -> 3 -> 6 -> 7 -> 10 -> 11
path 5:	1 -> 2 -> 5 -> 6 -> 7 -> 10 -> 11
path 6:	1 -> 2 -> 3 -> 6 -> 9 -> 10 -> 11
path 7:	1 -> 2 -> 5 -> 6 -> 9 -> 10 -> 11
path 8:	1 -> 2 -> 3 -> 6 -> 7 -> 10 -> 12 -> 13 -> 14
path 9:	1 -> 2 -> 5 -> 6 -> 7 -> 10 -> 12 -> 13 -> 14
path 10:	1 -> 2 -> 3 -> 6 -> 9 -> 10 -> 12 -> 13 -> 14
path 11:	1 -> 2 -> 5 -> 6 -> 7 -> 10 -> 12 -> 13 -> 14
path 12:	1 -> 2 -> 3 -> 6 -> 7 -> 10 -> 12 -> 13 -> 15 -> 16
path 13:	1 -> 2 -> 5 -> 6 -> 7 -> 10 -> 12 -> 13 -> 15 -> 16
path 14:	1 -> 2 -> 3 -> 6 -> 9 -> 10 -> 12 -> 13 -> 15 -> 16
path 15:	1 -> 2 -> 5 -> 6 -> 9 -> 10 -> 12 -> 13 -> 15 -> 16

The number of paths can be enormous in a large model. The SURE computer program automatically finds all the paths in the model.

Path-Step Classification

Once a particular path has been isolated for analysis, White's theorem is easily applied. Each state along the path must first be classified into one of three classes. These classes are distinguished by the type of transitions leaving the state. A state and the transitions leaving it will be referred to as a "path step." The transition on the path currently being analyzed will be referred to as the "on-path transition." The remaining transitions will be referred to as the "off-path transitions." The classification is made on the basis of whether the on-path and off-path transitions are slow (and hence also exponential) or fast. If there are no off-path transitions, the path step is classified as if it contained a slow off-path transition. Thus, the following classes of path steps are of interest.

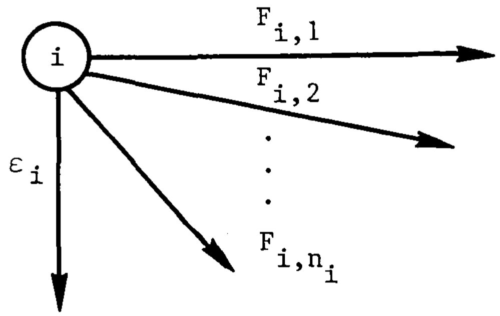
Class 1: slow on path, slow off path.-



Sketch A

The rate of the on-path exponential transition is λ_i. (See sketch A.) There may be an arbitrary number of slow off-path transitions. The sum of their exponential transition rates is γ_i. If any of the off-path transitions are not slow, then the path step is in class 3 described subsequently. The path steps 1 -> 2 and 5 -> 6 in the SIFT model (fig. 2) are examples.

Class 2: fast on path, arbitrary off path.-



Sketch B

There may be an arbitrary number of slow or fast off-path transitions. As before, the off-path exponential transitions can be represented as a single transition with a rate equal to the sum of all the off-path transition rates ε_i. (See sketch B.) The path steps 2 -> 5 and 3 -> 6 in the SIFT model (fig. 2) are examples. The distribution of time for a fast transition from state i to state k is referred to as $F_{i,k}$. Three measurable parameters must be specified for each of the n_i fast transitions. These are the transition probability $\rho(F_{i,k})$, the conditional mean $\mu(F_{i,k})$, and the conditional variance $\sigma^2(F_{i,k})$, given that this transition occurs. Mathematically, these parameters are defined as follows:

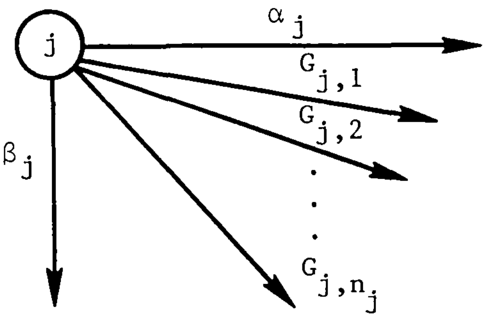
$$\rho(F_{i,k}) = \int_0^\infty \prod_{j\neq k} [1 - F_{i,j}(t)]\, dF_{i,k}(t)$$

$$\mu(F_{i,k}) = \frac{1}{\rho(F_{i,k})} \int_0^\infty t \prod_{j \neq k} [1 - F_{i,j}(t)]\, dF_{i,k}(t)$$

$$\sigma^2(F_{i,k}) = \frac{1}{\rho(F_{i,k})} \int_0^\infty t^2 \prod_{j \neq k} [1 - F_{i,j}(t)]\, dF_{i,k}(t) - \mu^2(F_{i,k})$$

Experimentally, these parameters correspond to the fraction of times that a transition is successful and the mean and variance of its recovery time distribution. It should be noted that these expressions are defined independently of the exponential transitions ε_j. Consequently, the sum of the fast off-path transition probabilities $\rho(F_{i,k})$ is 1. In particular, if there is only one recovery transition, its probability is 1 and the conditional mean is equivalent to the unconditional mean recovery time.

Class 3: slow on path, fast off path.-



Sketch C

This class includes path steps with both slow and fast off-path transitions. At least one off-path transition must be fast or the path step is in class 1. (See sketch C.) The on-path transition must be slow. The path steps 2 -> 3 and 7 -> 8 in the SIFT model (fig. 2) are in this class. The slow on-path transition rate is α_j. The sum of the slow off-path transition rates is β_j. As in class 2, the transition probability $\rho(G_{j,k})$, the conditional mean $\mu(G_{j,k})$, and the conditional variance $\sigma^2(G_{j,k})$ must be given for each fast off-path transition with distribution $G_{j,k}$. Although, the parameters described above suffice to specify a class 3 path step to SURE, the mathematical theory is more easily expressed in terms of the holding time in the state. Letting H_j represent the distribution of the holding time in state j, then the following parameters are used in White's theorem:

$$\mu(H_j) = \int_0^\infty \prod_{k=1}^{n_j} [1 - G_{j,k}(t)]\, dt$$

$$\sigma^2(H_j) = 2\int_0^\infty t \prod_{k=1}^{n_j} [1 - G_{j,k}(t)]\, dt - \mu^2(H_j)$$

These parameters are the mean and variance of the holding time in state j without consideration to the slow exponential transitions (i.e., with the slow exponential transitions removed). These parameters do not have to be explicitly given to the SURE program. The SURE program derives these parameters from the other available inputs — $\rho(G_{j,k})$, $\mu(G_{j,k})$, and $\sigma^2(G_{j,k})$ — as follows:

$$\mu(H_j) = \sum_{k=1}^{n_j} \rho(G_{j,k})\, \mu(G_{j,k})$$

$$\sigma^2(H_j) = \left\{ \sum_{k=1}^{n_j} \rho(G_{j,k}) \left[\sigma^2(G_{j,k}) + \mu^2(G_{j,k})\right] \right\} - \mu^2(H_j)$$

Although, the recovery time distributions are specified without consideration of the competing slow exponential transitions, White's theorem gives bounds that are correct in the presence of such exponential transitions. The parameters were defined in this manner to simplify the process of specifying a model. Throughout the paper, the holding time in a state in which the slow transitions have been removed will be referred to as "recovery holding time".

For convenience, when referring to a specific path in the model, an on-path recovery distribution will be indicated by using only one subscript indicating the source state. For example, if the transition with distribution $F_{j,k}$ is the on-path transition, then it can be referred to as F_j:

$F_{j,k}$ = the kth recovery transition from state j

F_j = the on-path recovery transition from state j (in a class 2 path step)

White's Multiple Recovery Theorem

With the above classification, White's theorem can now be given. The probability $D(T)$ of entering a particular death state within the mission time T, following a path with k class 1 path steps, m class 2 path steps, and n class 3 path steps, is bounded as follows:

$$LB \leqq D(T) \leqq UB$$

where

$$UB = E(T) \prod_{i=1}^{m} \rho(F_i) \prod_{j=1}^{n} \alpha_j \, \mu(H_j)$$

$$LB = E(T - \Delta) \prod_{i=1}^{m} \rho(F_i) \left[1 - \varepsilon_i \, \mu(F_i) - \frac{\mu^2(F_i) + \sigma^2(F_i)}{r_i^2} \right] \prod_{j=1}^{n} \alpha_j \left\{ \mu(H_j) - \frac{(\alpha_j + \beta_j)[\mu^2(H_j) + \sigma^2(H_j)]}{2} - \frac{\mu^2(H_j) + \sigma^2(H_j)}{s_j} \right\}$$

for all values of $r_i > 0$, $s_j > 0$, and

$$\Delta = r_1 + \ldots + r_m + s_1 + \ldots + s_n$$

and

E(T) = the probability of traversing a path consisting of only the class 1 path steps within time T.

The theorem is true for any $r_i > 0$ and $s_j > 0$. Different choices of these parameters will lead to different bounds. A mathematical procedure for selecting optimal values of r_i and s_j (i.e., leading to the closest bounds) has not been developed. The SURE program uses the following values of r_i and s_j:

$$r_i = \kappa_i [\mu(F_i) + \sigma(F_i)]$$

$$s_j = \kappa'_j [\mu(H_j) + \sigma(H_j)]$$

where κ_i and κ'_j are either automatically set (by default) to

$$\kappa_i = \mu^{1/2}(F_i) \Big/ [\mu(F_i) + \sigma(F_i)]$$

$$\kappa'_j = \mu^{1/2}(H_j) \Big/ [\mu(H_j) + \sigma(H_j)]$$

or are set equal to a user-chosen constant by the LBFACT command. For example, if LBFACT = 20, then κ_i and κ'_j are equal to 20 for all i and j. The default values of κ_i and κ'_j correspond to the values used in the original version of SURE. The default values have been found to give very close bounds in practice, usually very near the optimal choice. Of course, the SURE user can experiment with the LBFACT to obtain closer bounds if desired. (Note that the default values of κ_i and κ'_j are not dimensionless. Therefore, the lower bound may show a slight variation with changes in units of measure. This will not occur if the LBFACT command is used.)

Two simple algebraic approximations for E(T) were given by White (ref. 2) — one that overestimates and one that underestimates, respectively:

$$E_u(T) = \frac{\lambda_1\lambda_2\lambda_3 \cdots \lambda_k T^k}{k!} > E(T)$$

$$E_\ell(T) = E_u(T)\left[1 - \frac{T}{k+1}\sum_{i=1}^{k}(\lambda_i + \gamma_i)\right] < E(T)$$

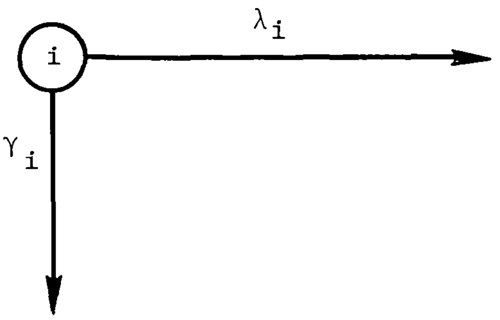
Both $E_u(T)$ and $E_\ell(T)$ are close to E(T) as long as $T \sum (\lambda_i + \gamma_i)$ is small. The SURE program uses these approximations unless the ETCALC=1 option is invoked. When the ETCALC=1 option is used, a differential equation solver is used to calculate E(T) and E(T - Δ). This option is discussed in a subsequent section entitled "The SURE User Interface".

BOUNDS BASED ON MEANS AND PERCENTILES

Path-Step Classification

The path-step classification defined in the previous section is also useful for describing Lee's technique (ref. 4). The primary difference (from a user's perspective) between Lee's technique and White's technique is the method of describing the fast recovery transitions. Lee's technique utilizes the transition probabilities, the conditional mean holding time, as well as a user-chosen percentile of the recovery holding time distribution. The required information for each class is listed below:

Class 1: slow on path, slow off path.-

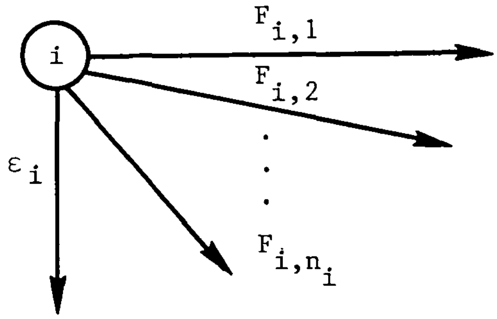


Sketch A (repeated)

λ_i = the on-path exponential transition rate

γ_i = the sum of the off-path exponential transition rates

Class 2: fast on path, arbitrary off path.-



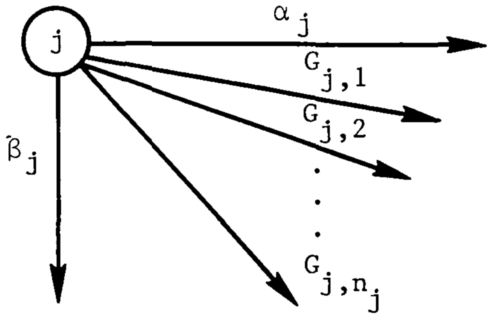
Sketch B (repeated)

$$\rho(F_{i,k}) = \int_0^\infty \prod_{j\neq k} [1 - F_{i,j}(t)]\, dF_{i,k}(t)$$

= the probability that the on-path transition from state i to state k is successful (mathematically and experimentally the same as in White's theory)

ε_i = the sum of the off-path exponential transition rates

Class 3: slow on path, fast off path.-



Sketch C (repeated)

α_j = the slow on-path transition rate

$\rho(G_{j,k})$ = the probability that transition from state j to state k is successful. (This must be specified for all the competing off-path recovery transitions.)

ξ_j = the percentile point of the distribution chosen by the user. This can also be viewed as the censoring point of an experiment (i.e., the longest time the experimenter waits for a transition to occur).

$H_j(\xi_j)$ = the probability that the recovery holding time (i.e., with slow transitions removed) in state j is less than ξ_j

$\mu(\xi_j)$ = the conditional mean recovery holding time in the source state j, given that the fastest transition occurs before ξ_j,

$$= \int_0^{\xi_j} t \, dH_j(t) \Big/ H_j(\xi_j), \text{ where } H_j(t) = 1 - \prod_{k=1}^{n_j} [1 - G_{j,k}(t)]$$

β_j = the sum of the slow off-path transition rates

Using the above notation, Lee's theorem is easily stated in the following discussion.

Lee's Multiple Recovery Theorem

The probability $D(T)$ of entering a particular death state within the mission time T, following a path with k class 1 path steps, m class 2 path steps, and n class 3 path steps, is bounded as follows:

$$LB \leqq D(T) \leqq UB$$

where

$$UB = E(T) \prod_{i=1}^{m} \rho(F_i) \prod_{j=1}^{n} \alpha_j \{H_j(\xi_j)\, \mu(\xi_j) + [1 - H_j(\xi_j)][\xi_j + 1/(\alpha_j + \beta_j)]\}$$

$$LB = E(T - \Delta) \prod_{i=1}^{m} [\exp(-\varepsilon_i \xi_i)][\rho(F_i) + H_i(\xi_i) - 1] \prod_{j=1}^{n} \alpha_j \exp[-(\alpha_j + \beta_j)\xi_j]\{H_j(\xi_j)\, \mu(\xi_j) + \xi_j[1 - H_j(\xi_j)]\}$$

$$\Delta = \sum_{i=1}^{m} \xi_i + \sum_{j=1}^{n} \xi_j$$

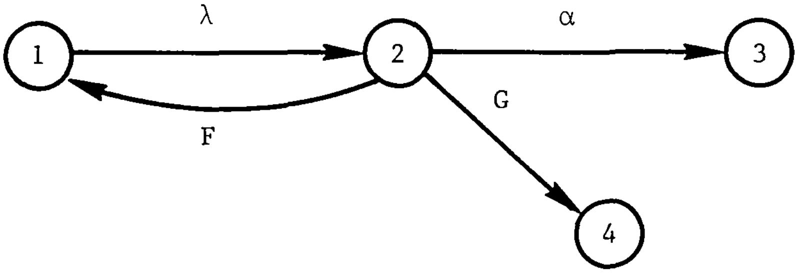
and

$E(T)$ = the probability of traversing a path consisting of only the class 1 path steps within time T

The user of the SURE program is free to use either White's method or Lee's method. In many ways the choice is merely a matter of taste. White's method appears to be more convenient for design studies in which properties of the recovery distributions are assumed. Engineering judgment appears to be more skillful at predicting means and percentiles. Lee's method is especially adapted for the analysis of models for which experimental data is available. This method explicitly takes into consideration the problem of censored data. However, it is clear that either Lee's or White's methods could be used for both design studies and experimental analyses.

TRANSIENT AND INTERMITTENT MODELS

The mathematical techniques developed by White and Lee do not explicitly accommodate semi-Markov models that are not pure death processes. The problem with nonpure death process models is that the circuits in the graph structure of the model lead to an infinite sequence of paths of increasing length. Consider the following semi-Markov model (see sketch D) of a system susceptible to transient faults:



Sketch D

The parameter λ is the arrival rate of transient faults in the system. The duration of a transient fault is described by the distribution function $F(t)$, which competes with a system reconfiguration process with distribution $G(t)$. The circuit in this model leads to an infinite sequence of paths:

```
1 -> 2 -> 3
1 -> 2 -> 1 -> 2 -> 3
1 -> 2 -> 1 -> 2 -> 1 -> 2 -> 3
1 -> 2 -> 1 -> 2 -> 1 -> 2 -> 1 -> 2 -> 3
  .
  .
  .
```

However, the longer the path the less significant its contribution to the probability of entering death state 3. If we let $P_3^{(k)}(T)$ be the probability of being in state 3 at time T after traversing the loop k times, then using White's theorem gives

$$P_3^{(k)}(T) = \alpha \; \mu(H) \; \rho^k(F) \; (\lambda T)^{k+1} / (k + 1)!$$

where $\mu(H)$ is the mean recovery holding time in state 2. Then, the probability of being in state 3 at time T (by any path) is

$$P_3(T) = \sum_{k=0}^{\infty} P_3^{(k)}(T) = (\alpha/\rho)\ \mu(H)[\exp(\lambda\rho T) - 1]$$

where $\rho = \rho(F)$. The infinite series above converges to an exponential function. The convergence of this series is very fast for $\lambda\rho T < 1$. Since λ is the rate of a slow transition, this relationship holds. Accurate values can be obtained using only two or three terms of the series. In general, the error in truncating the series after n terms (using Taylor's theorem) is less than

$$\exp(\lambda\rho T)\ (\lambda\rho T)^{n+1}/(n + 1)!$$

The SURE program automatically unfolds a circuit into a sequence of paths. The truncation point is user-specifiable via the TRUNC or PRUNE command. If

```
TRUNC = 4
```

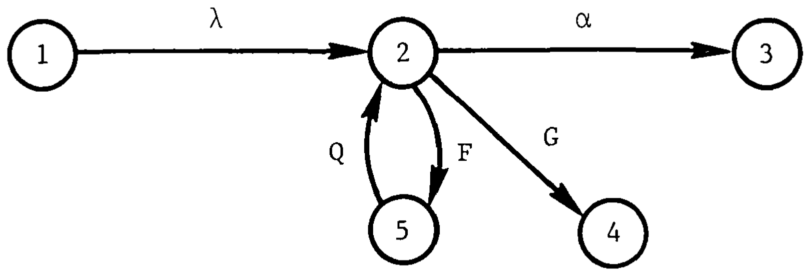
then the sequence of paths is terminated after unfolding the loop four times. This is equivalent to truncating the series after the fourth term. It is recommended that the user try several values of TRUNC until convergence is certain. The PRUNE command may also be used to terminate the unfolding process after the probability of the path falls below the value of the PRUNE constant. For example,

```
PRUNE = 1E-18;
```

will terminate the processing of all paths whose probability drops below 10^{-18}. The PRUNE constant also specifies the termination of nonloop paths that fall below the specified probability level. (See the subsequent section entitled "Pruning".) If it is desired that "pruning" be performed exclusively, the TRUNC constant should be set to a large value, e.g., 10 000.

In order to use SURE, the transition probabilities $\rho(F)$ and $\rho(G)$ must either be calculated from some known distribution or be measured experimentally. Similarly, the mean and variance of the holding time in state 2 must be determined.

Intermittent faults can be modeled in a similar manner. The major difference is that an intermittent fault does not totally disappear. (See sketch E.) Therefore, a benign state with holding time $Q(t)$ is introduced in the model to represent the fault while it is not active:



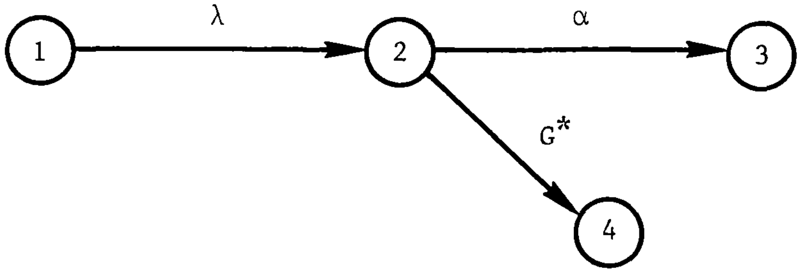
Sketch E

Computationally, however, the problem is different. Since the circuit in sketch E contains only fast transitions, the rate of convergence can be very slow. Using White's upper bound gives

$$P_3(t) \approx \alpha\lambda\ \mu(C_2)\ T[1 + \rho(F) + \rho^2(F) + \rho^3(F) + \ldots]$$

For simplicity, suppose that F and G are exponentially distributed. Then, $\rho(F) = \mu(G)/[\mu(F) + \mu(G)]$. The percentage error in truncating the series after n terms is easily shown to be $[\rho(F)]^{n+1} \times 100\%$. If $\rho(F)$ is near 1 (i.e., when $\mu(G)$ is large relative to $\mu(F)$), then n must be large to get the percentage error acceptably low. For example, if $\mu(F) = 10$ and $\mu(G) = 1000$, then n (and thus TRUNC) must be equal to 302 in order to have a percentage error of 5 percent.

An alternative approach is recommended when convergence is slow. The above model can be collapsed into the following model (see sketch F):



Sketch F

The new recovery transition G* contains the percentage of the intermittent fault. Experimentally, G* represents the time to recover in the presence of the intermittent fault. If experimental data is not available, but a parametric form is available for the distributions F, G, and Q, then the mean and variance of G* can be calculated. For example, if F, G, and Q are exponential, then

$$\mu(G^*) = [\mu(Q) + \mu(F)]\ [\mu(G)/\mu(F)]$$

$$\sigma^2(G*) = \left[\frac{\mu(Q)\ \mu(G)}{\mu(F)} + \mu(G)\right]^2 + \frac{2\mu^2(Q)\ \mu(G)}{\mu(F)}$$

In this section, a general proof that convergence will always be obtained for arbitrary models with arbitrary loops has not been given. The analysis used in this section can be applied to many similar models to demonstrate that convergence will occur. However, a general proof has not been attempted because of the large number of cases involved. Nevertheless, the SURE program provides a solution technique for models with loops if convergence occurs. The lack of convergence can be observed by increasing the TRUNC constant. A model for which the upper bound does not converge has not yet been found.

THE SURE USER INTERFACE

Basic Program Concept

The user of the SURE program must describe his semi-Markov model to the SURE program by using a simple language for enumerating all the transitions of the model. This language will be discussed in detail in this section.

The SURE user must first assign numbers to every state in the system. The semi-Markov model is then described by enumerating all the transitions. As described in the previous sections, each transition is classified as being either slow or fast. Consequently, there are two different statements used to enter transitions — one for slow transitions and the other for fast. If a transition is slow, then the following kind of statement is used:

```
1,2 = 0.0001;
```

This defines a slow exponential transition from state 1 to state 2 with rate 0.0001. (The program does not require any particular units, e.g., hour^{-1} or sec^{-1}. However, the user must be consistent.) If the transition is fast, then either of two methods can be used to describe the transition. These methods correspond to White's and Lee's methods discussed previously. The following specifies a fast (recovery) transition using White's method:

```
2,4 = < 1E-4, 1E-6, 1.0 >;
```

The numbers in the brackets correspond to the conditional mean, conditional standard deviation, and transition probability of the fast transition, respectively. The syntax of the transition-description statements is described more fully in the following section.

SURE Model Definition Syntax

The transition-description statements described above are the only essential ingredients in the SURE language. However, the flexibility of the SURE program has been increased by adding several features commonly seen in programming languages such as FORTRAN or Pascal. The details of the SURE language are described in the following subsections.

Lexical details.- The state numbers must be positive integers between 0 and the MAXSTATE implementation limit, usually 10 000. (This limit can be changed by redefining a constant in the SURE program and recompiling the SURE source.) The transition rates, conditional means and standard deviations, etc., are floating point numbers. The Pascal REAL syntax is used for these numbers. Thus, all the following would be accepted by the SURE program:

```
0.001
12.34
1.2E-4
1E-5
```

The semicolon is used for statement termination. Therefore, more than one statement may be entered on a line. Comments may be included any place that blanks are allowed. The notation "(*" indicates the beginning of a comment and "*)" indicates the termination of a comment. The following is an example of the use of a comment:

```
LAMBDA = 5.7E-4;    (* FAILURE RATE OF A PROCESSOR *)
```

If statements are entered from a terminal (as opposed to by the READ command described below), then the carriage return is interpreted as a semicolon. Thus, interactive statements do not have to be terminated by an explicit semicolon unless more than one statement is entered on the line.

In interactive mode the SURE system will prompt the user for input by a line number followed by a question mark. For example,

```
1?
```

The number is a count of the syntactically correct lines entered into the system thus far plus the current one.

Constant definitions.- The user may equate numbers to identifiers. Thereafter, these constant identifiers may be used instead of the numbers. For example,

```
LAMBDA = 0.0052;
RECOVER = 0.005;
```

Constants may also be defined in terms of previously defined constants:

```
GAMMA = 10*LAMBDA;
```

In general, the syntax is

```
"name" = "expression";
```

where "name" is a string of up to eight letters, digits, and underscores (_) beginning with a letter, and "expression" is an arbitrary mathematical expression as described in a subsequent section entitled "Expressions".

Variable definition.- In order to facilitate "parametric analyses", a single variable may be defined. A range is given for this variable. The SURE system will compute the system reliability as a function of this variable. If the system is run in graphics mode (to be described later), then a plot of this function will be made. The following statement defines LAMBDA as a variable with range 0.001 to 0.009:

```
LAMBDA = 0.001 TO 0.009;
```

Only one such variable may be defined. A special constant, POINTS, defines the number of points over this range to be computed. The method used to vary the variable over this range can be either geometric or arithmetic and is best explained by example. Thus, suppose POINTS = 4, then

Geometric:

```
XV = 1 TO* 1000;
```

where the values of XV used would be 1, 10, 100, and 1000.

Arithmetic:

```
XV = 1 TO+ 1000;
```

where the values of XV used would be 1, 333, 667, and 1000.

The * following the TO implies a geometric range. A + or nothing following the TO implies an arithmetic range.

One additional option is available — the BY option. By following the above syntax with BY "inc", the value of POINTS is automatically set such that the range is varied by the specified amount. For example,

```
V = 1E-6 TO* 1E-2 BY 10;
```

sets POINTS equal to 5, and the values of V used would be 1E-6, 1E-5, 1E-4, 1E-3, and 1E-2. The statement

```
Q = 3 TO+ 5 BY 1;
```

sets POINTS equal to 3, and the values of Q used would be 3, 4, and 5.

In general, the syntax is

```
"var" = "expression" TO "c" "expression" { BY "inc" }
```

where "var" is a string of up to eight letters and digits beginning with a letter, "expression" is an arbitrary mathematical expression as described in the next section entitled "Expressions", and "c" is a +, *, or nothing. The BY clause is optional; but if it is used, then "inc" is an arbitrary expression.

Expressions.- When specifying transition or holding time parameters in a statement, arbitrary functions of the constants and the variable may be used. The following operators may be used:

+	addition
-	subtraction
*	multiplication
/	division
**	exponentiation

The following standard functions may be used:

EXP(X)	exponential function
LN(X)	natural logarithm
SIN(X)	sine function
COS(X)	cosine function
ARCSIN(X)	arc sine function
ARCCOS(X)	arc cosine function
ARCTAN(X)	arc tangent function
SQRT(X)	square root

Both ( ) and [ ] may be used for grouping in the expressions. The following are permissible expressions:

```
2E-4
1.2*EXP(-3*ALPHA);
7*ALPHA + 12*LAMBDA;
ALPHA*(1+LAMBDA) + ALPHA**2;
2*LAMBDA + (1/ALPHA)*[LAMBDA + (1/ALPHA)];
```

Slow-transition description.- A slow transition is completely specified by citing the source state, the destination state, and the transition rate. The syntax is as follows:

```
"source" , "dest" = "rate";
```

where "source" is the source state, "dest" is the destination state, and "rate" is any valid expression defining the exponential rate of the transition. The following are valid SURE statements:

```
PERM = 1E-4;
TRANSIENT = 10*PERM;

1,2 = 5*PERM;
1,9 = 5*(TRANSIENT + PERM);
2,3 = 1E-6;
```

In the notation of the theory section we have

$$i,j = \lambda_i;$$

Fast-transition description.- To enter a fast transition, the SURE user may use either of two methods — White's method or Lee's method — described in the subsequent discussion.

White's method: The following syntax is used for White's method.

```
"source" , "dest"  = <"mu", "sig", "frac">;
```

where

"mu" = an expression defining the conditional mean transition time, $\mu(F)$

"sig" = an expression defining the conditional standard deviation of the transition time, $\sigma(F)$

"frac" = an expression defining the transition probability, $\rho(F)$

and "source" and "dest" define the source and destination states, respectively. In the notation of the theory section, we have

$$i,j = < \mu(F_i), \sigma(F_i), \rho(F_i) >;$$

The third parameter "frac" is optional. If omitted, the transition probability is assumed to be 1.0, i.e., only one recovery transition. All the following are valid (while in White's mode):

```
2,5 = <1E-5, 1E-6, 0.9>;

THETA = 1E-4;
5,7 = <THETA, THETA*THETA, 0.5>;
7,9 = <0.0001,THETA/25>;
```

Lee's method: To describe a fast transition using Lee's method, the following syntax is used.

```
"source" , "dest" =  < "frac" >;

@ "source"  =  <"hmu", "quant", "prob">;
```

where

"source" = source state

"dest" = destination state

"frac" = an expression defining the transition probability, $\rho(F)$

"hmu" = an expression defining the conditional mean recovery holding time in the state $\mu(\xi)$, given that holding time is less than "quant"

"quant" = an expression defining the percentile or censoring point, ξ

"prob" = an expression defining the probability $H(\xi)$, that the holding time in the state is less than "quant"

In the notation of the theory section we have

$$j,k = < \rho(F_j) >;$$

$$@j = < \mu(\xi_j),\ \xi_j,\ H_j(\xi_j) >;$$

All the following are valid SURE statements (while in the Lee mode):

```
5,6 = <0.5>;
FRACT = 0.0 TO 0.5;
5,7 = <FRACT>;
5,8 = <0.5 - FRACT>;

@5 = < 0.00034, 0.003, (1.0-1E-4) >;
```

Although there may be many recovery transitions from a state, the "@ source" statement must be issued only once for the source state.

The SURE user must decide which method he will use before entering his model. Either the Lee method or White method may be used to describe the model, but both cannot be used at the same time. By default, the program assumes that the White method will be used. If Lee's method is desired, the LEE command must be issued prior to entering any recovery transition.

SURE Command Syntax

READ command.- A sequence of SURE statements may be read from a disk file. The following interactive command reads SURE statements from a disk file named SIFT.MOD:

```
READ SIFT.MOD;
```

If no file name extent is given, the default extent .MOD is assumed. A user can build a model description file using a text editor and use this command to read it into the SURE program.

INPUT command.- This command increases the flexibility of the READ command. Within the model description file created with a text editor, INPUT commands can be inserted that will prompt for values of specified constants while the model file is being read. For example, the command

```
INPUT LVAL;
```

will prompt the user for a number as follows:

```
LVAL?
```

and a new constant LVAL is created that is equal to the value input by the user. Several constants can be interactively defined as in the following example:

```
INPUT X, Y, Z;
```

SHOW command.- The value of an identifier may be displayed by the following command:

```
SHOW ALPHA;
```

Information about a transition may also be displayed by the SHOW command. For example, information concerning the transition from state 654 to state 193 is displayed by the following command:

```
        SHOW 654-193;
```

Information about a state holding time may be displayed. For example, state 12 holding time characteristics are listed in response to

```
        SHOW 12;
```

More than one transition, identifier, or holding time may be shown at one time:

```
        SHOW ALPHA, 12-13, BETA, 123;
```

CALC command.- For convenience, a calculator function has been included. This command allows the user to obtain the value of an arbitrary expression. For example,

```
        X = 1.6E-1;
        CALC  (X-.12)*EXP(-0.001) + X**3;
              = 4.405601999335E-02
```

LEE command.- The LEE command prepares the program to receive recovery transition commands according to the Lee syntax. By default, the program expects recovery transitions to be described in White format. The syntax of the LEE command is

```
        LEE;
```

The LEE command must be issued prior to entering any recovery transitions.

LIST options.- The amount of information output by the program is controlled by this command. Four list modes are available as follows:

LIST = 0; No output is sent to the terminal, but the results can still be displayed using the PLOT command.

LIST = 1; Only the upper and lower bounds on the probability of total system failure are listed. This is the default.

LIST = 2; The probability bounds for each death state in the model are reported along with the totals.

LIST = 3; Every path in the model is listed and its probability of traversal. The probability bounds for each death state in the model is reported along with the totals.

RUN command.- After a semi-Markov model has been fully described to the SURE program, the RUN command is used to initiate the computation:

```
RUN;
```

The output is displayed on the terminal according to the LIST option specified. If the user wants the output written to a disk file instead, the following syntax is used:

```
RUN "outname";
```

where the output file "outname" may be any permissible VAX VMS file name.

EXIT command.- The EXIT command causes termination of the SURE program.

HELP command. - The HELP command generates a brief description of all the SURE commands.

Graphics display commands. - If the appropriate graphics hardware/software is available, then SURE generates graphical displays of the reliability models and plots the upper and lower bounds of the total system probability of failure as a function of a single variable. The user indicates by joystick input where each state of the model should be displayed. The user must issue the DISP command

```
DISP;
```

prior to the transition commands to cause the system to prompt for the state locations. The system automatically pans as the model exceeds the current scope of the screen. Once the user indicates where each state should be placed, the program automatically draws all the transitions and labels them. The user may retain the state location information on disk using the SAVE command. For example, the current state location information is written to file SIFT.MEG by the following command:

```
SAVE SIFT.MEG
```

State location information may be retrieved from a disk file by using the GET command. If state location has been stored on disk file FTMP.MEG in a prior SURE session, then the following command will retrieve this information:

```
GET FTMP.MEG
```

If the location information is on a file with the same VMS file name (except the extent) as the command file that describes the model, then the following is an abbreviation for the commands GET TRIPLEX.MEG; READ TRIPLEX.MOD:

```
READ TRIPLEX*;
```

The extent names must be .MOD for the file containing the model commands and .MEG for the file containing the state locations on the graphics display. The SCAN and ZOOM commands may be used to peruse the model. The joystick button is used to end the ZOOM and SCAN commands.

PLOT command.- After a RUN or CALC command, the PLOT command can be used to plot the output on the graphics display. The syntax is

```
PLOT <op>, <op>, ... <op>
```

where <op> are plot options. Any TEMPLATE "USET" or "UPSET" parameter can be used, but the following are the most useful:

XLOG plot x-axis using logarithmic scale
YLOG plot y-axis using logarithmic scale
XYLOG plot both x- and y-axes using logarithmic scales
NOLO plot x- and y-axes with normal scaling

XLEN=5.0 set x-axis length to 5.0 in.
YLEN=8.0 set y-axis length to 8.0 in.
XMIN=2.0 set x-origin 2 in. from left side of screen
YMIN=2.0 set y-origin 2 in. above bottom of screen

Special constants.- Certain user-defined "constants" are interpreted by the SURE program in a special way. These are described as follows:

TIME = 100; sets the mission time to 100. The default TIME is 10.

POINTS = 100; indicates that 100 points should be calculated/plotted over the range of the variable. Default value is 25.

ETCALC = 1; directs program to use a differential equation solver to calculate E(T) rather than the algebraic approximations. This is useful when the mission time is long. The default is ETCALC = 0, which selects White's algebraic formulas for E(T).

PRUNE = 1E-12; sets pruning level to 1E-12. The program will stop searching a path after its current probability becomes less than the specified level. The default is PRUNE = 0.0.

WARNDIG = 2; sets the number of digits accuracy desired in the upper bound to 2 when using the PRUNE capability. The default is 1. If the pruning process could have caused the upper bound to be less accurate than specified by this constant, the warning message

PRUNING TOO SEVERE

is given. See next section entitled "Pruning".

TRUNC = 2; sets truncation point at 2. Default is 3. If a circuit (i.e., infinite loop) is found in the graph, this determines the maximum number of times that the loop will be traversed.

LBFACT = 20; sets the κ_i and κ'_j constants in White's theorem to 20.

ECHO = 0; turns off the echo when reading a disk file. The default value of ECHO is 1, which causes the model description to be listed as it is read. (See example 3 in a subsequent section entitled "Example SURE Sessions.")

START = 15; indicates that start state is 15. Default is 1.

Pruning.- The time required to analyze a large model can often be greatly reduced by model pruning. It is essential that this be done carefully in order to maintain accuracy. The SURE user specifies the depth of pruning desired using the PRUNE constant. A path is traversed by the SURE program until the probability of reaching the current point on the path falls below the pruning level. Clearly, the probability of reaching a death state by continuing along this path must be less than the pruning level. The error resulting from this pruning method is therefore less than the product of the number of paths pruned (NPP) and the value of the PRUNE constant. The SURE program will warn the user if this product is great enough to lead to less than WARNDIG digits of accuracy. In other words, a warning message PRUNING TOO SEVERE is generated if

$$(NPP)(PRUNE) > P_f / 10^{WARNDIG}$$

where P_f is the upper bound on system failure. This warning message is very conservative. Typically, the accuracy is far greater than is guaranteed by this test.

EXAMPLE SURE SESSIONS

Outline of a Typical Session

The SURE program was designed for interactive use. The following method of use is recommended:

1. Create a file of SURE commands using a text editor describing the semi-Markov model to be analyzed.

2. Start the SURE program and use the READ command to retrieve the model information from this file.

3. Then, miscellaneous commands may be used to change the values of the special constants, such as LIST, POINTS, ETCALC, TRUNC, etc., as desired. However, altering the value of a constant identifier does not affect any transitions entered previously even though they were defined when using the constant. The range of the variable may be changed after transitions are entered.

4. Enter the RUN command to initiate the computation.

Examples

The following examples illustrate interactive SURE sessions. For clarity, all user inputs are given in lower-case letters.

Example 1.- This session illustrates direct interactive input and the type of error message given by SURE:

```
$ sure

  SURE V2.3    NASA Langley Research Center

  1? lambda = 1e-5;
  2? 1,2 = 6*lambda;
  3? 2,3 = 5*lamba;
                    ^ IDENTIFIER NOT DEFINED
  3? 2,3 = 5*lambda;
  4? show 2-3;
 TRANSITION 2 -> 3: RATE = 5.00000E-5
  5? 2,4 = <1e-4,1e-5>;
  6? 4,5 = 2*lambda;
  7? list = 2;
  8? time = 10;
  9? run

 DEATHSTATE     LOWERBOUND    UPPERBOUND
 ----------     ----------    ----------
        3       2.96584E-12   3.00000E-12
        5       5.98581E-08   6.00000E-08

 TOTAL          5.98610E-08   6.00030E-08

 *** WARNING: SYNTAX ERRORS PRESENT BEFORE RUN
 2 PATH(S) PROCESSED
 0.060 SECS. CPU TIME UTILIZED

 10? exit
```

The warning message is simply informative. If a user receives this message, he should check his input file to make sure that the model description is correct. In this example, since the syntax error was corrected in the next line, the model was correct. A complete list of computer generated error messages is given in the appendix.

Example 2.- The following session indicates the normal method of using SURE. Prior to this session, a text editor has been used to build file TRIADP1.MOD, and TRIADP1.MEG was created by the SAVE command in a previous session.

```
$ sure

  SURE V2.3    NASA Langley Research Center

  1? read triadp1*;

  2: LAMBDA = 1E-6 TO* 1E-2;
  3: RECOVER = 2.7E-4;
  4: STDEV = 1.3E-3;
  5: 1,2 = 3*LAMBDA;
  6: 2,3 = 2*LAMBDA;
  7: 2,4 = <RECOVER,STDEV>;
  8: 4,5 = 3*LAMBDA;
  9: 5,6 = 2*LAMBDA;
 10: 5,7 = <RECOVER,STDEV>;
 11: 7,8 = LAMBDA;
 12: POINTS = 10;
 13: TIME = 6;

 14? run

    LAMBDA        LOWERBOUND     UPPERBOUND
  -----------    -----------    -----------
  1.00000E-06    6.15619E-15    1.00441E-14
  2.78256E-06    5.20064E-14    8.22407E-14
  7.74264E-06    4.96189E-13    7.33127E-13
  2.15444E-05    5.85630E-12    7.75250E-12
  5.99484E-05    8.87204E-11    1.04754E-10
  1.66810E-04    1.62004E-09    1.77475E-09
  4.64159E-04    3.25617E-08    3.45029E-08
  1.29155E-03    6.78293E-07    7.14440E-07
  3.59382E-03    1.41278E-05    1.51683E-05
  1.00000E-02    2.82284E-04    3.25059E-04

  3 PATH(S) PROCESSED
  0.320 SECS. CPU TIME UTILIZED

15? plot ylog
16? exit
```

Figure 3 illustrates the model displayed on the output graphics device (defined in file TRIADP1.MEG). The plot in figure 4 was generated from this run by the plot ylog command.

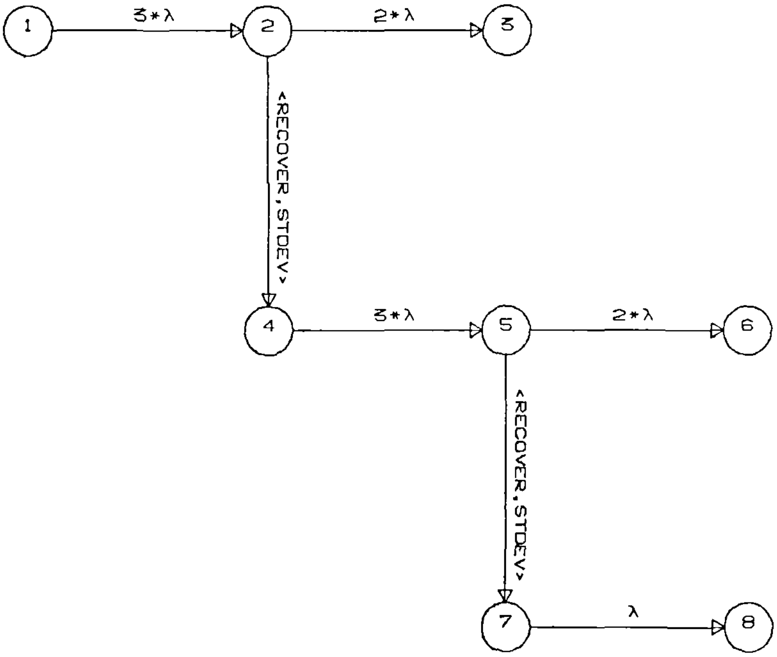


Figure 3.- Semi-Markov model from example 2.

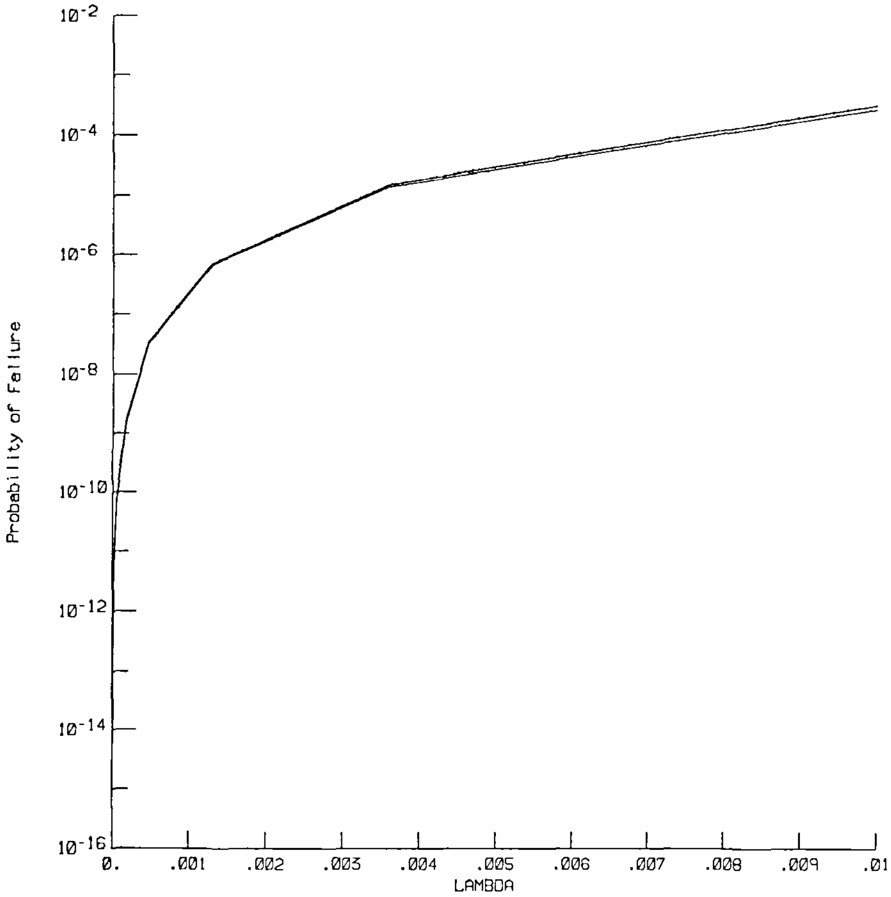


Figure 4.- SURE output from example 2.

Example 3.- The following interactive session illustrates the use of the ECHO constant. This constant is used when the model description file is large and one desires that the model input not be listed on the terminal as it is read by the SURE program.

```
$sure

  SURE V2.3    NASA Langley Research Center

  1? get ftmp.meg
  2? echo = 0;
  3? read ftmp2.mod;

 26? points = 1;
 *** POINTS   CHANGED TO 1.00000E+00
 27? run

    LAMBDA        LOWERBOUND     UPPERBOUND
  -----------    -----------    -----------
  1.00000E-04    4.58240E-10    5.02254E-10


 28? exit
```

Example 4.- This interactive session illustrates how SURE can be used to obtain system unreliability as a function of mission time.

```
$ sure

  SURE V2.3    NASA Langley Research Center

  1? read ftmp9

  2: LAMBDA = 5E-4;              (* PERMANENT FAULT RATE *)
  3: STDEV = 3.6E-4;             (* STAN. DEV. OF RECOVERY DISTRIBUTION *)
  4: RECOVER = 2.7E-4;           (* MEAN OF RECOVERY DISTRIBUTION *)
  5: TIME = 0.1 TO* 1000 BY 10;
  6: 1,2 = 9*LAMBDA;
  7: 2,3 = 2*LAMBDA;
  8: 2,4 = <RECOVER,STDEV>;
  9: 4,5 = 9*LAMBDA;
 10: 5,6 = 2*LAMBDA;
 11: 5,7 = <RECOVER,STDEV>;
 12: 7,8 = 6*LAMBDA;
 13: 8,9 = 2*LAMBDA;
 14: 8,10 = <RECOVER,STDEV>;
 15: 10,11 = 6*LAMBDA;
 16: 11,12 = 2*LAMBDA;
 17: 11,13 = <RECOVER,STDEV>;
 18: 13,14 = 6*LAMBDA;
 19: 14,15 = 2*LAMBDA;
 20: 14,16 = <RECOVER,STDEV>;
 21: 16,17 = 3*LAMBDA;
 22: 17,18 = 2*LAMBDA;
 23: 17,19 = <RECOVER,STDEV>;
 24: 19,20 = 1*LAMBDA;
 25: START = 1;

 26? run

    TIME          LOWERBOUND      UPPERBOUND
  -----------    -----------    -----------
  1.00000E-01    9.68946E-11    1.21527E-10
  1.00000E+00    1.14040E-09    1.21774E-09
  1.00000E+01    1.15701E-08    1.24261E-08
  1.00000E+02    1.16095E-07    1.59925E-07
  1.00000E+03    0.00000E+00    8.13719E-02 .. E(T) INACCURATE

 7 PATH(S) PROCESSED
 0.390 SECS. CPU TIME UTILIZED

 27? exit
```

Since the E(T) INACCURATE message was obtained, the computation should be rerun with ETCALC=1:

```
$ sure

  SURE V2.3    NASA Langley Research Center

  1? echo = 0; read ftmp9;

 26? etcalc = 1;
 27? run

   TIME           LOWERBOUND     UPPERBOUND
  -----------    -----------    -----------
  1.00000E-01    9.68946E-11    1.21500E-10
  1.00000E+00    1.14040E-09    1.21500E-09
  1.00000E+01    1.15741E-08    1.21487E-08
  1.00000E+02    1.21138E-07    1.26748E-07
  1.00000E+03    7.66073E-03    7.69898E-03

 7 PATH(S) PROCESSED
 28.180 SECS. CPU TIME UTILIZED

 28? exit
```

Example 5.- This example illustrates the use of SURE to solve a model of a triplex system with transient and permanent faults. A transient fault model is not a pure death process, and consequently there is a loop in the graph. Therefore, it is necessary to use the PRUNE or TRUNC feature of SURE. In this example, the TRUNC feature is used:

```
$ sure

  SURE V2.3   NASA Langley Research Center

  1? read 3trans*

  2: LAMBDA = 1E-4;              (* FAULT ARRIVAL RATE *)
  3: INPUT DELTA;                (* MEAN RECOVERY TIME *)
     DELTA? 1800
  4: GAMMA = 10*LAMBDA;          (* TRANSIENT FAULT RATE *)
  5: RHO = 1 TO* 1E7 BY 10;      (* RECOVERY RATE FROM TRANSIENT FAULT *)
  6: PHI = DELTA;                (* RATE TRANSIENTS RECONFIGURED OUT *)
  7: T = RHO + DELTA;
  8:
  9: 1,2 = 3*LAMBDA;
 10: 2,3 = 2*LAMBDA + 2*GAMMA;
 11: 2,4 = <1/DELTA,1/DELTA,1.0>;
 12: 4,5 = LAMBDA + GAMMA;
 13: 1,6 = 3*GAMMA;
 14: 6,1 = <1/T,1/T,RHO/T>;
 15: 6,4 = <1/T,1/T,PHI/T>;
 16: 6,7 = 2*LAMBDA + 2*GAMMA;

 17? run

 START STATE ASSUMED TO BE 1

    RHO           LOWERBOUND    UPPERBOUND
  -----------    -----------   -----------
  1.00000E+00    1.77753E-04   1.81450E-04
  1.00000E+01    1.76961E-04   1.80639E-04
  1.00000E+02    1.69449E-04   1.72945E-04
  1.00000E+03    1.20664E-04   1.23038E-04
  1.00000E+04    4.12696E-05   4.20342E-05
  1.00000E+05    1.92279E-05   1.96140E-05
  1.00000E+06    1.66239E-05   1.69692E-05
  1.00000E+07    1.63587E-05   1.66999E-05

  12 PATH(S) PROCESSED
  1 PATH(S) TRUNCATED AT DEPTH 3
  3.020 SECS. CPU TIME UTILIZED

 18? plot xylog
 19? exit
```

Figure 5 illustrates the model displayed on the output graphics device. (Note that the Greek words in the model description are displayed as Greek characters on the graphics output). The plot in figure 6 was generated from this run.

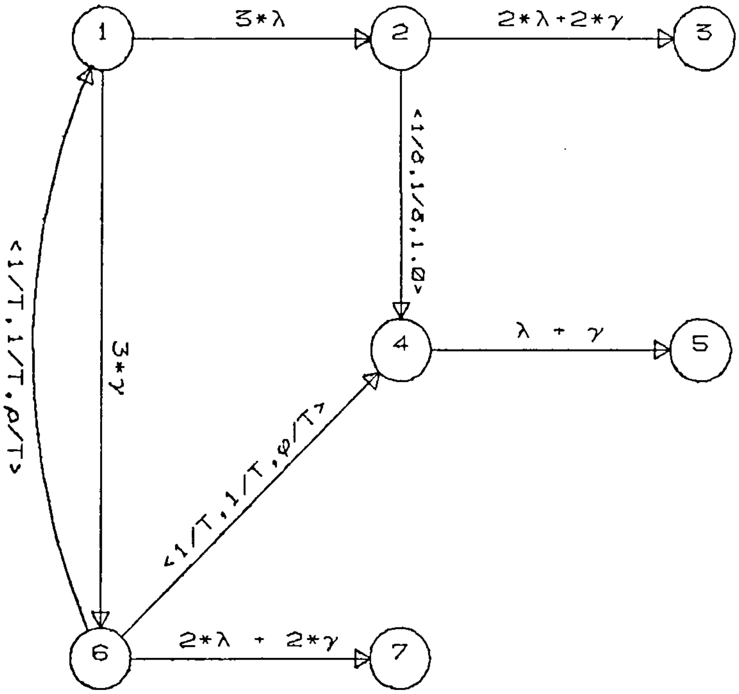


Figure 5.- Semi-Markov model from example 5.

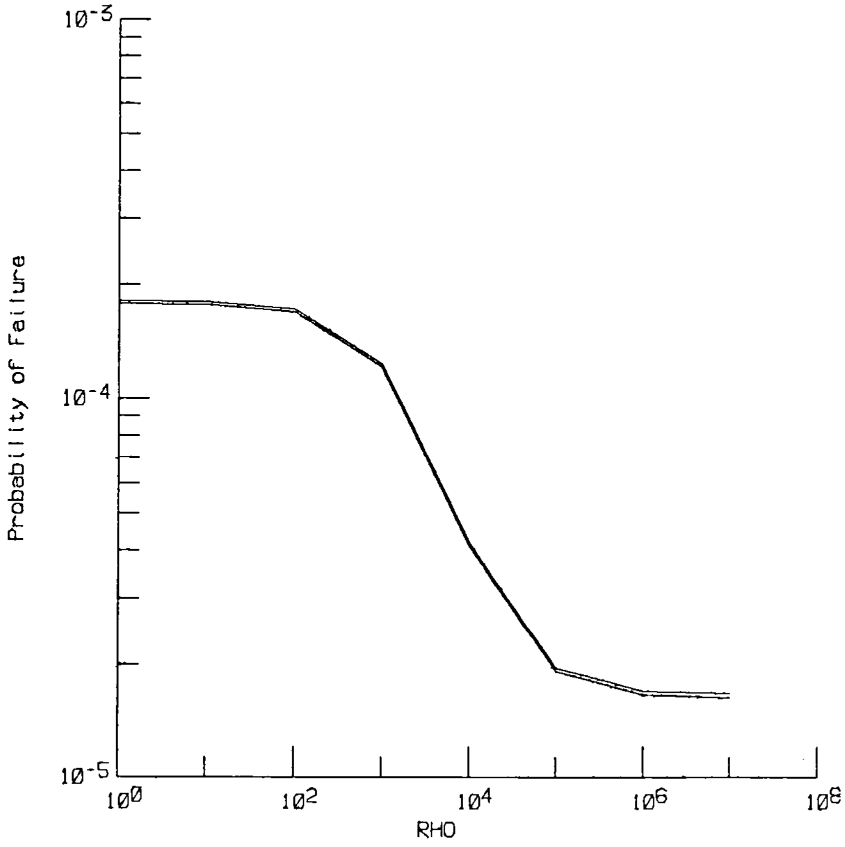


Figure 6.- SURE output from example 5.

Example 6.- This example illustrates the use of SURE in Lee mode. The same model used in example 5 is used here.

```
$ sure

  SURE V2.3    NASA Langley Research Center

  1? read leem

  2: LEE;
  3: LAMBDA = 1E-4;            (* FAULT ARRIVAL RATE *)
  4: DELTA = 1800.0;           (* MEAN RECOVERY TIME *0
  5: GAMMA = 10*LAMBDA;        (* TRANSIENT FAULT RATE *)
  6: RHO = 1 TO* 1E7 BY 10;    (* RECOVERY RATE FROM TRANSIENT FAULT *)
  7: PHI = DELTA;              (* RATE TRANSIENTS RECONFIGURED OUT *)
  8: T = RHO + PHI;
  9: QUANT2 = 1E-2;
 10: QPROB2 = 1.0 - EXP(-DELTA*QUANT2);
 11: TIME = 10;
 12: 1,2 = 3*LAMBDA;
 13: 2,3 = 2*LAMBDA + 2*GAMMA;
 14: @2 = <1/DELTA, QUANT2, QPROB2>;
 15: 2,4 = <1.0>;
 16: 4,5 = LAMBDA + GAMMA;
 17: 1,6 = 3*GAMMA;
 18: QUANT6 = 1E-2;
 19: QPROB6 = 1.0 - EXP(-T*QUANT6);
 20: @6 = <1/T,QUANT6,QPROB6>;
 21: 6,1 = <RHO/T>;
 22: 6,4 = <PHI/T>;
 23: 6,7 = 2*LAMBDA + 2*GAMMA;

 24? run

 ----- LEE STATISTICAL ANALYSIS MODE -----

    RHO           LOWERBOUND     UPPERBOUND
  -----------    -----------    -----------
  1.00000E+00    1.78430E-04    1.81450E-04
  1.00000E+01    1.77632E-04    1.80639E-04
  1.00000E+02    1.70066E-04    1.72945E-04
  1.00000E+03    1.20986E-04    1.23038E-04
  1.00000E+04    4.13316E-05    4.20343E-05
  1.00000E+05    1.92859E-05    1.96141E-05
  1.00000E+06    1.66853E-05    1.69692E-05
  1.00000E+07    1.64206E-05    1.67000E-05

 12 PATH(S) PROCESSED
 1 PATH(S) TRUNCATED AT DEPTH 3
 2.200 SECS. CPU TIME UTILIZED
```

Example 7.- This example illustrates the use of Lee's method to model a system with two possible recoveries from a fault. In this model, the system recovers from a fault by bringing in a (nonfailed) spare 90 percent of the time and degrades to a simplex 10 percent of the time.

```
$ sure

  SURE V2.3    NASA Langley Research Center

  1? lee;
  2? lambda = 1e-4;       (* Failure rate of a processor *)
  3? pr1 = 0.90;          (* Probability recovery is by sparing *)
  4? mu = 2e-4;           (* Mean recovery time *)
  5? 1,2 = 3*lambda;
  6? 2,3 = 2*lambda;
  7? 2,4 = <pr1>;
  8? 4,5 = 3*lambda;
  9? 5,6 = 2*lambda;
 10? 2,7 = <1-pr1>;
 11? 7,8 = lambda;
 12? @2 = <mu,2*mu,1.0>;   (* No observed recoveries greater than 2*MU *)
 13? list=2;
 14? run

 ----- LEE STATISTICAL ANALYSIS MODE -----

 DEATHSTATE   LOWERBOUND    UPPERBOUND
 ----------   -----------   -----------
      3       1.19815E-10   1.20000E-10
      6       2.69428E-09   2.70000E-09
      8       1.49788E-07   1.50000E-07

  TOTAL       1.52602E-07   1.52820E-07

 3 PATH(S) PROCESSED
 0.040 SECS. CPU TIME UTILIZED
```

CONCLUDING REMARKS

The SURE program is a flexible, user-friendly, interactive design/validation tool. The program provides a rapid computational capability for semi-Markov models useful in describing the fault-handling behavior of fault-tolerant computer systems. The only modeling restriction imposed by the program is that general recovery transitions must be fast in comparison to the mission time. For systems with recovery times greater than the mission time, the bounds are still correct, but they are not close together. The SURE reliability analysis method utilizes a fast approximation theory developed by Allan L. White of PRC Kentron, Inc., and later generalized by Larry D. Lee of the Langley Research Center and White. This approximation theory enables the calculation of upper and lower bounds on system reliability. These upper and lower bounds are typically within about 5 percent of each other. Since the computation method is extremely fast, large state spaces are not a problem. Therefore, state aggregation techniques are not utilized.

Although the approximation theory does not explicitly deal with semi-Markov models that are not pure death processes, the SURE program utilizes simple path truncation strategies to enable the analysis of such models. Consequently, transient and intermittent behavior of fault-tolerant computer systems can be investigated with the SURE program.

NASA Langley Research Center
Hampton, VA 23665-5225
November 1, 1985

APPENDIX

ERROR MESSAGES

The following error messages are generated by the SURE system. These are listed in alphabetical order:

ARGUMENT TO EXP FUNCTION MUST BE < 8.80289E+01 - The argument to the EXP function is too large.

ARGUMENT TO LN OR SQRT MUST BE > 0 - The LN and SQRT function require positive arguments.

ARGUMENT TO STANDARD FUNCTION MISSING - No argument was supplied for a standard function.

COMMA EXPECTED - Syntax error; a comma is needed.

CONSTANT EXPECTED - Syntax error; a constant is expected.

DIVISION BY ZERO NOT ALLOWED - A division by 0 was encountered when evaluating the expression.

ERROR OPENING FILE - <vms status> - The SURE system was unable to open the indicated file.

E(T) INACCURATE - The entered mission time is so large that it makes the upper and lower bounds very far apart.

FILE NAME TOO LONG - File names must be 80 or less characters.

FILE NAME EXPECTED - Syntax error; the file name is missing.

ID NOT FOUND - The system is unable to SHOW the identifier since it has not yet been defined.

IDENTIFIER EXPECTED - Syntax error; identifier expected here.

IDENTIFIER NOT DEFINED - The identifier entered has not yet been defined.

ILLEGAL CHARACTER - The character used is not recognized by SURE.

ILLEGAL STATEMENT - The command word is unknown by the system.

INPUT LINE TOO LONG - The command line exceeds the 100 character limit.

INTEGER EXPECTED - Syntax error; an integer is expected.

LEE @ REQUIRES THREE PARAMETERS - The @ statement requires three parameters in the LEE mode.

MUST BE IN "READ" MODE - The INPUT command can be used only in a file processed by a READ command.

NUMBER TOO LONG - Only 15 digits/characters allowed per number.

ONLY 1 VARIABLE ALLOWED - Only one variable can be defined per model.

REAL EXPECTED - A floating point number is expected here.

RECOVERY TOO SLOW - A recovery transition in the model is too slow to permit close upper and lower bounds.

SEMICOLON EXPECTED - Syntax error; a semicolon is needed.

STATE OUT OF RANGE - The state number is negative or greater than the maximum state limit (Default = 10 000, set at SURE compilation time).

ST. DEV TOO BIG - The standard deviation of a recovery distribution is too large to permit close upper and lower bounds.

SUB-EXPRESSION TOO LARGE, i.e. > 1.70000E+38 - An overflow condition was encouraged when evaluating the expression.

THIS CONSTRUCT NOT PERMITTED IN LEE MODE - This construct is not allowed while in the LEE mode.

THIS CONSTRUCT NOT PERMITTED IN WHITE MODE - This construct is not allowed while in the WHITE mode.

TRANSITION NOT FOUND - The system is unable to SHOW the transition because it has not yet been defined.

VMS FILE NOT FOUND - The file indicated on the READ command is not present on the disk. (Note: make sure your default directory is correct.)

WARNING: VARIABLE CHANGED! - If previous transitions have been defined using a variable and the variable name is changed, inconsistencies can result in the values of the transitions.

WARNING: MORE THAN ONE STARTSTATE - The model entered by the user has more than one start state (i.e., a state with no transitions to it).

*** WARNING: SYNTAX ERRORS PRESENT BEFORE RUN - Syntax errors were present during the model description process.

*** WARNING: RUN-PROCESSING ERRORS - Computation overflow occurred during execution.

*** ID CHANGED TO X - The value of the identifier (Constant) is being changed.

*** ID CHANGED TO X TO Y - The range of the variable is being changed.

***** TRANSITION X -> Y ALREADY ENTERED - The user is attempting to reenter the same transition again.

= EXPECTED - Syntax error; the = operator is needed.

> EXPECTED - Syntax error; the closing bracket > is missing.

) EXPECTED - A right parenthesis is missing in the expression.

] EXPECTED - A right bracket is missing in the expression.

REFERENCES

1. Geist, Robert M.; and Trivedi, Kishor S.: Ultrahigh Reliability Prediction in Fault-Tolerant Computer Systems. IEEE Trans. Comput., C-32, no. 12, Dec. 1983, pp. 1118-1127.

2. White, Allan L.: Upper and Lower Bounds for Semi-Markov Reliability Models of Reconfigurable Systems. NASA CR-172340, 1984.

3. Butler, Ricky W.: The Semi-Markov Unreliability Range Evaluator (SURE) Program. NASA TM-86261, 1984.

4. Lee, Larry D.: Reliability Bounds for Fault-Tolerant Systems With Competing Responses to Component Failures. NASA TP-2409, 1985.

5. White, Allan L.: Synthetic Bounds for Semi-Markov Reliability Models. NASA CR-178008, 1985.

6. Lala, Jaynarayan H.; and Smith, T. Basil, III: Development and Evaluation of a Fault-Tolerant Multiprocessor (FTMP) Computer. Volume III - FTMP Test and Evaluation. NASA CR-166073, 1983.

7. Trivedi, Kishor; Dugan, Joanne Bechta; Geist, Robert; and Smotherman, Mark: Modeling Imperfect Coverage in Fault-Tolerant Systems. The Fourteenth International Conference on Fault-Tolerant Computing - FTCS 14, Digest of Papers, 84CH2050-3, IEEE, 1984, pp. 77-82.

8. Bavuso, S. J.; and Petersen, P. L.: CARE III Model Overview and User's Guide (First Revision). NASA TM-86404, 1985.

9. Goldberg, Jack; Kautz, William H.; Melliar-Smith, P. Michael; Green, Milton W.; Levitt, Karl N.; Schwartz, Richard L.; and Weinstock, Charles B.: Development and Analysis of the Software Implemented Fault-Tolerance (SIFT) Computer. NASA CR-172146, 1984.

1. Report No.
NASA TM-87593

2. Government Accession No.

3. Recipient's Catalog No.

4. Title and Subtitle

The SURE Reliability Analysis Program

5. Report Date
February 1986

6. Performing Organization Code
505-34-13-32

7. Author(s)
Ricky W. Butler

8. Performing Organization Report No.
L-16005

9. Performing Organization Name and Address

NASA Langley Research Center
Hampton, VA 23665-5225

10. Work Unit No.

11. Contract or Grant No.

12. Sponsoring Agency Name and Address
National Aeronautics and Space Administration
Washington, DC 20546-0001

13. Type of Report and Period Covered
Technical Memorandum

14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The SURE program is a new reliability analysis tool for ultrareliable computer system architectures. The program is based on computational methods recently developed for the NASA Langley Research Center. These methods provide an efficient means for computing accurate upper and lower bounds for the death-state probabilities of a large class of semi-Markov models. Once a semi-Markov model is described using a simple input language, the SURE program automatically computes the upper and lower bounds on the probability of system failure. A parameter of the model can be specified as a variable over a range of values directing the SURE program to perform a sensitivity analysis automatically. This feature, along with the speed of the program, makes it especially useful as a design tool.

17. Key Words (Suggested by Author(s))
Reliability analysis
Semi-Markov models
Fault tolerance
Reliability modeling
Markov models

18. Distribution Statement
Unclassified - Unlimited

Subject Category 65

19. Security Classif. (of this report)
Unclassified

20. Security Classif. (of this page)
Unclassified

21. No. of Pages
43

22. Price
A03

For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA-Langley, 1986